Graphics • C • Pascal • ASM • BASIC • 386

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal™

MORE OBJECT-ORIENTED GRAMMING

November/December 1989
Volume 7.6
$4.95 Canada $5.95

A WINDOWS TERMINAL PROGRAM

MANAGING INTERFACES WITH CURSES

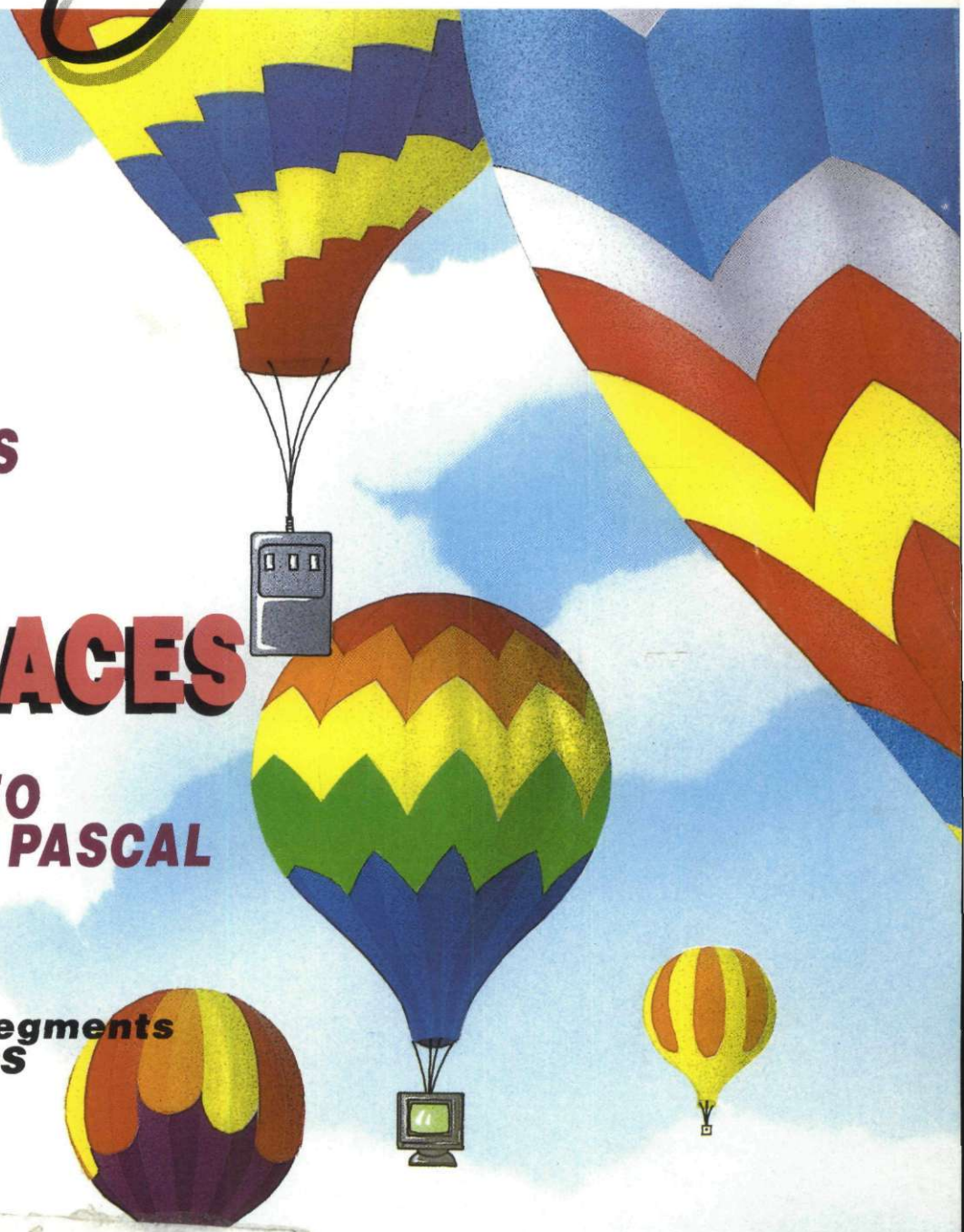
USER INTERFACES

STEREO VIDEO WITH TURBO PASCAL

VGA and EGA Split Screens

Four-Gigabyte Segments in Real-Mode DOS

80X86 Memory Models



RON K UNZ
WALL STREET ANALYTICS INC
33-41 85TH ST
JACKSON HEIGHTS NY 11372

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Now it's easy to make your programs millions of times smarter.

More and more, programmers and work station builders are using DESQview as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview —multitasking, windowing, intertask communications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

Bells and Whistles

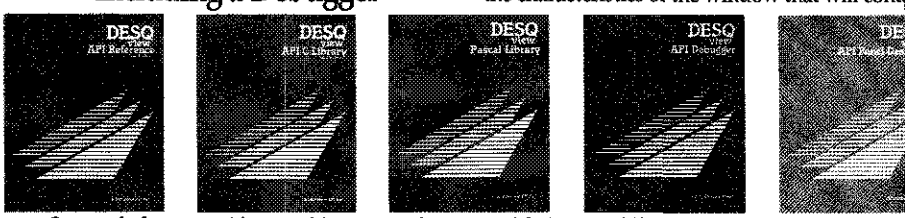
A program taking advantage of the DESQview API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data; or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

Some of the applications under development right now using DESQview API Tools: CAD, medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.

80386 Power

Programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview—side by side with other 80386 and DOS programs. The breakthroughs that make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders.

A Complete Interactive Toolkit —Including a Debugger



Quarterdeck can provide everything you need to put DESQview capabilities into your applications.

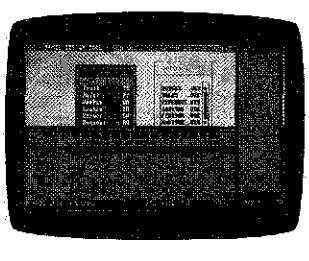
API Reference Manual The key to the power of the DESQview API, this manual contains all you need to know to write Assembly Language programs that take full advantage of DESQview's capabilities. And there's an 'include' file with symbols and macros to aid you in development.

API C Library Here are C language interfaces for the entire set of API functions. It supports the Lattice™ C, Metaware™ C, Microsoft® C, and Turbo C compilers for all memory models. Included with the C Library package is the API Reference Manual and source code for the library.

API Pascal Library The Pascal library provides interfaces for the entire set of API functions. It supports Turbo Pascal V4.0 and V5.0 compilers. Included are the API Reference Manual, source code for the library, and example programs.

API Debugger The DESQview API Debugger is an interactive tool enabling the API programmer to trace and single step through API calls from several concurrently running DESQview-specific programs. Trace information is reported symbolically along with the program counter,registers, and stack at the time of the call. Trace conditions can be specified to report only calls of interest

API Panel Designer This interactive tool h you design windows, menus, help screens, erro messages, and forms. It includes an editor that you construct an image of your panel using sim commands to enter, edit, copy, and move text, a well as draw lines and boxes. You can then defi the characteristics of the window that will cont the panel, such as its position, size, and title. Finally, you can specify the locations and types fields in the panel.



The Panel Desig automatically generates all the DESQview API streams necessar display and take input from your panel. These dat streams may be grouped into panel libraries and stored on disk as part of your program.

More Tools are Coming Quarterdeck is comm to adding tools as needed by our users. To that we have been working with Ashton Tate and B words International on dBASE III and dBASE I translators. And in the works, we have BASIC a DOS Extender libraries.

If you'd like hands-on experience with the API Toolkit, write for information on our next API Conference, coming this August.

Bring New Power to DOS with the DESQview API Toolkit

Quarterdeck

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-38

Number **147** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"Mary had a 4GL who's performance was very slow and everywhere that Mary went the run-times had to go."

PRO-C
The C Source Code Applications Generator

Accelerate Your Productivity

Pro-C gives you the greatest gift in the computer world--time. You get high quality, fully commented, error free C source code in a fraction of the time it would take to write it by hand.

Pro-C will increase your ability to create programs quickly by generating the source code for menus, reports, screens, windows, and multi-file batch updates. Textbook quality C code is written by us, while the elegant system solutions and exciting new algorithms are created by you.

Pro-C looks and feels like a 4GL, but its not. You can do prototyping, layout, design and generation of applications without learning a proprietary language, needing massive amounts of memory, or ending up with slow running programs. Best of all Pro-C doesn't require any run-time licenses. Finally, a company that treats you like royalty instead of forcing you to pay them.

Pro-C--the programming partner that does the boring, repetitive coding without complaint, pays for itself every time you use it, and doesn't argue with your obviously brilliant program designs.

PRO-C Order yours to-day. Call 1-800-265-2682 VESTRONIX

Number 287 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal™

Volume 7.6
November/December 1989

Contents

4 Notes from the Editor

6 Letters

12 Consulting: Wealth Through Project Growth *by William H. Roetzheim*
How to expand your existing consulting work into larger contracts

14 On Graphics: VGA and EGA Split Screens *by Michael Abrash*
Using the nifty but surprisingly tricky split-screen features of the EGA and VGA

26 User Interface: A Simple Terminal Program for Windows *by William S. Hall*
Our Windows/PM expert illustrates using the communications interface along with font selection and dialog boxes under Microsoft Windows

40 User Interface: Human Interfaces with Curses *by Vaughn Vernon*
Text-based user interfaces under DOS and UNIX with the Curses C library for device independence

50 User Interface: Cooking with Stereo Video *by Guy Cole*
Using two video displays in your next Turbo Pascal application

63 ASM: Efficient 80X86 Memory Models and Static Variables *by John Otken*
Using static and automatic variables, equates, and macros for efficient assembly language programming

73 C: Aztec C and the ANSI C Standard Delayed *by Rex Jaeschke*
Rex reports on the delays in getting out the ANSI C standard, along with a look at Manx's Aztec C version 4.2

78 Pascal: Using OOP to Create a Powerful File Viewer *by Tom Swan*
Tom illustrates the value of object-oriented programming in Turbo Pascal 5.5 by creating a powerful file utility

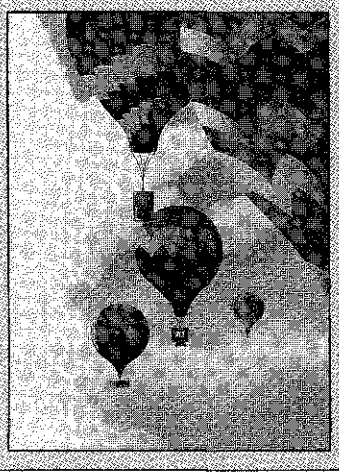
89 386 Now: Four Gigabytes in Real Mode *by Thomas Roden*
A slick trick to access large memory spaces on the 80386 from DOS

95 BASIC: QuickBASIC's Fast String-Searching Algorithm *by Ethan Winer*
QuickBASIC contains a fast string-searching algorithm that may be useful to other-language programmers as well

100 Miscellaneous *by M. Steven Baker*
Prototyping/User Interface Tools
Proteus (Helios)
Mini-Case Tools
Logic Gem (Sterling Castle)
Books
System BIOS for IBM PC/XT/AT series
by Phoenix Technologies (Addison-Wesley)

102 *PJ* Billboard

104 Advertiser Index



User interfaces already come in many forms, shapes, and sizes—and even a few different colors. New user interfaces are being launched every month, and while some never get off the ground, others take flight with great success. This issue of PJ shows you how to use various interfaces to help you rise above the programming pack.

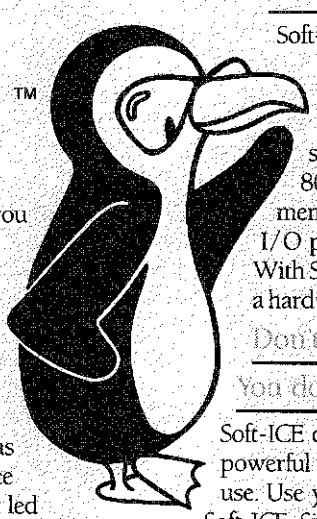
Cover Artist:
Jarrett Jester

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY. A debugging tool tough enough to handle the DOS Nasties.

New Version 2.0

Soft-ICE™



Nasty over-write? No sweat!

Soft-ICE memory range break points help you track down memory over-write problems whether you are doing the over-writing or another program is over-writing you.

Hung program? No problem!

When the system hangs, you now have hope. With Soft-ICE you can break out of hung programs no matter how bad the system has been trashed. And with Soft-ICE's back trace ranges you can re-play the instructions that led up to the crash.

Program too large? Not with Soft-ICE!

Soft-ICE runs entirely in extended memory. This means you can debug even the largest DOS programs. And since your program runs at the same address whether Soft-ICE is loaded or not you can find those subtle bugs that change when the starting address of your code changes.

System debugging? Soft-ICE is a natural!

Soft-ICE is ideal for full source level debugging of TSRs, interrupt service routines, self booting programs, DOS loadable device drivers, real-time kernels, non-DOS O/Ss and ROMs. Soft-ICE can even debug within DOS & BIOS.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses the 80386 to provide real-time break points on memory locations, memory ranges, execution, I/O ports, hardware & software interrupts. With Soft-ICE you get all the speed and power of a hardware-assisted debugger at a software price.

Don't want to switch debuggers? You don't have to!

Soft-ICE can run stand-alone or it can add its powerful break points to the debugger you already use. Use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated automatically.

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

"These may be the only two products I've seen in the last two or three years that exceeded my wildest expectations for power, compatibility and ease-of-use."

—Paul Mace
Paul Mace Software

Soft-ICE	$386
MagicCV	$199
MagicCV for Windows	$199
Buy Soft-ICE & MagicCV(W)	—Save $86.
Buy MagicCV and MagicCVW	—Save $100.
Buy All 3	—Save $186.

30 day money-back guarantee
Visa, MasterCard and AmEx accepted

1988 BYTE Award of Distinction

New Soft-ICE 2.0 features

- Back Trace Ranges
- Symbolic & Source level debugging
- EMS 4.0 support with special EMS debugging commands
- Windowed user interface

Nu-Mega™ TECHNOLOGIES

CALL TODAY (603) 888-2386
or FAX (603) 888-2465

RUN CODEVIEW IN 8K

MagicCV™

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView in less than 8K of conventional memory on your 80386 PC.

NEW—Version 2.0 includes EMS 4.0 driver.
Attention Windows Developers!
Version available for CVW.

P.O. BOX 7607 ■ NASHUA, NH ■ 03060-7607

Number **160** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal™

Editor
M. Steven Baker
Managing Editor
Robert E. Wanetick
Contributing Editors
Michael Abrash
Frank D. Greco
Tom Swan

Contributing Writers
Michael Abrash
M. Steven Baker
Guy Cole
William S. Hall
Rex Jaeschke
John Otken
Thomas Roden
William H. Roetzheim
Tom Swan
Vaughn Vernon
Ethan Winer

Publisher
Elizabeth Oakley
Associate Publisher
Bobbi Sinyard
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Andrea McHone
Subscriptions
Gary Hoyer
Accounting
Lori Fanslow
Deborah Loftis
Administrative Assistant
Sylvia Long
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
503-747-0800
Fax: 503-746-0071

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in *Programmer's Journal* represent the views of the authors and not necessarily those of the publisher. Letters to the editor become the property of *Programmer's Journal* and may be edited for publication.

Code and listings in *Programmer's Journal* appear "as is," and *Programmer's Journal* makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. *Programmer's Journal* assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Andrea McHone at 503-747-0800, or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced using the following software products: Ventura Publisher,™ Aldus PC PageMaker,® and Corel Draw. PostScript™ files are output on a Linotronic™ 200.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

POSTMASTER: Forwarding and address correction requested. Send address changes to *Programmer's Journal*, P.O. Box 30160, Eugene, OR 97403.

© 1989 *OAKLEY PUBLISHING COMPANY*

NOTES from the Editor

After too many years, I finally made time for a real vacation. While I sit above the rim of Chaco Canyon in New Mexico gazing on crumbling Anasazi ruins, the computer world seems so far away. But magazine schedules still must be met.

This issue of *PJ* focuses on user interface design. If any facet of the microcomputer revolution could benefit from a distant view, it's the design of the user interface. The ease with which a new software product can be learned and used has become enormously more important in the marketplace. Less than a decade ago, in the olden days of CP/M, the command-line interface was most common. Of course, CPUs were 8 bits wide and memory limited to 64K. Command lines gave way to popular menu-driven interfaces. Consider the success of the Lotus-style menu bar in the PC world. Most of these earlier implementations of the user interface were text based. The resolution of early PC graphics adapters (CGAs) was barely tolerable for working with text, and implementing graphics on the PC meted out a substantial performance penalty.

Based on Xerox technology, the Apple Macintosh arrived on the mass market sporting a graphical user interface—the now famous Windows, Icon, Mouse, and Pull-down (WIMP) menu metaphor. The older Macintosh machines suffered performance problems because the 68000 CPU handled most of these graphics and I/O tasks. At least on IBM PC-based machines, a market has always existed for fast, smart peripheral devices such as high-resolution video adapters using the Texas Instruments TMS34010, the Intel 8514, and soon the likes of the new Intel i860. Nowadays, many PC applications sport a pull-down menu and pop-up interface based on concepts from the graphical user-interface world. These interface designs are available in both text-based (Borland) and graphics-based (Microsoft Windows and Digital Research's GEM) interfaces for the PC. But the direction of current interface-design thinking is clearly aimed at graphics.

Unfortunately for the PC programmer, a splintered market exists out there. Competing WIMP interfaces include PC-based Windows, GEM, and OS/2 Presentation Manager (PM), the UNIX-based OSF Motif, Sun News, AT&T/SUN's Open Look interface, and others based on MIT's X Window system. These interface designs are made possible by the availability of faster CPUs and fast, high-resolution graphics adapters. Unfortunately, the application programming interfaces (APIs) can be widely divergent. In addition, the learning curve to gain proficiency in any one of these API systems is steep and long. A question still remains in my mind whether for many applications a text-based interface of similar design wouldn't be more than satisfactory.

Operating Systems . . . Back to the Future

By the end of September, IBM plans on delivering version 1.2 of OS/2 with Presentation Manager. The new enhancements are a fast B+ tree-file system and a better command shell using icons. The file system is necessitated at the high end to compete with UNIX SQL-servers in database applications. In Boca Raton, Florida, IBM staff and consultants have been working around the clock for the last two months to ensure meeting the delivery schedule with a bug-free product. Meanwhile Microsoft is working on a Windows 3.0 version to emulate much of PM. We can only hope that the APIs for these systems remain stable enough that most existing applications won't be broken.

Although Microsoft was misquoted in a September *Infoworld*, one of these days soon, Windows will need to add support for the Virtual Control Program Interface (VCPI) to allow running programs using DOS extenders. One can't write off the likes of Lotus 1-2-3 release 3.0 from the Windows market. Virtually all of the high-end PC CAD market will soon belong to 386 versions using DOS extenders. Of course, DESQview conveniently supports VCPI now.

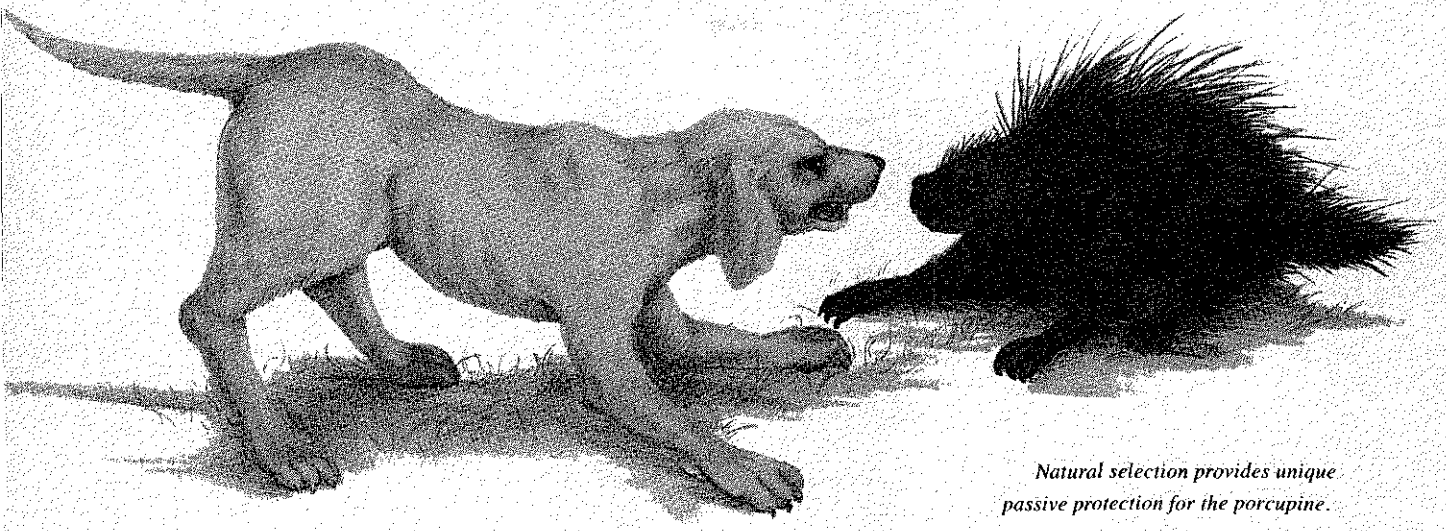
What Next . . . 386, 486, 586

As COMDEX approaches, a large number of vendors are working on 486 systems. The early 80486 chip was amazingly clean. Meanwhile, fine software products continue to arrive regularly. WATCOM (Waterloo, Ontario) is shipping their 80386 C compiler-and-development system for DOS extenders. Silicon Valley Software (San Mateo, California) has ported their line of 80386 C, Pascal, FORTRAN, and BASIC compilers to the Phar Lap DOS Extender. Lahey has new versions of their FORTRAN compilers.

As the sun sets over Pueblo Bonito, I'll sign off. —*Steve*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Developers



Natural selection provides unique passive protection for the porcupine.

The Activator - Natural Selection For Software Protection

Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.

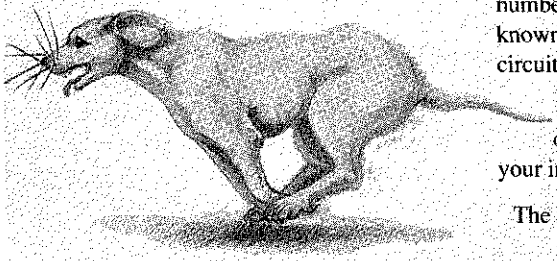
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

Programmable Option

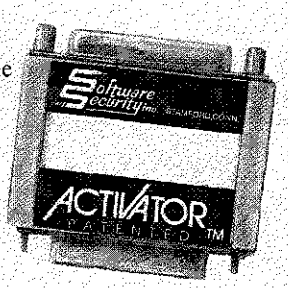
The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.



Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.

We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software."*

1-800-333-0407 ext.103
In Connecticut 203-329-8870
Fax 203-329-7428

Software Security inc.
1011 High Ridge Road
Stamford, CT 06905

Unlimited Copies • Programmable • No Batteries • Small Size • Fast • Patented • Data Protection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Poor Guideline

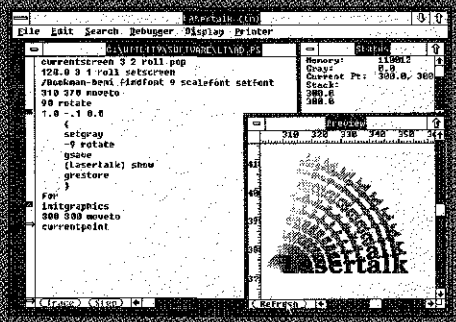
I enjoyed the article about coding conventions for assembly language programming, "A Line of Code," by John Otken, in *PJ* 7.4. This is the first article of its type that I have liked. Surprisingly, I agree one hundred percent with every point made by Mr. Otken—except one: He states that "a general guideline for register usage is to trash registers as necessary and save them as required when you write the calling procedure." In practice I have found this to be a poor guideline. It also is a waste of memory in many cases, especially for utility-like routines such as the STRLEN example in the article. My guidelines only allow for using the AX register without preserving it, and many times AX is a return value. When a utility routine is used more than once, space can be saved on every call, since any program of any complexity will be using most of the registers. Additionally, programmers tend to cut and paste simple routines from program to program. I've seen many bugs caused by the unknown side effects of registers changing values. (Of course, every programmer properly documents all such side effects.)

I must restate, however, that all of the other concepts described by Mr. Otken are almost identical to the style that I use. Thanks for the good article.

Mike Schmit
President, Quantasm Corporation
Cupertino, California

JOHN OTKEN RESPONDS: Oops, that "guideline" is unclear. There are three approaches to saving registers. The first is to save all registers that do not return values. This makes programs easier to write but slightly larger and slower—a good approach for small to medium programs. The second is to save only those registers in which calling routines keep important values. This approach executes faster but is harder to program and not very useful with library routines. The third approach lies between the first two: Save all registers except AX. I believe this is the approach described by Mr. Schmit. In virtually all cases, the push and pop instructions belong in the called procedure and not around the call instruction.

Master PostScript
Language Programming On Your PC

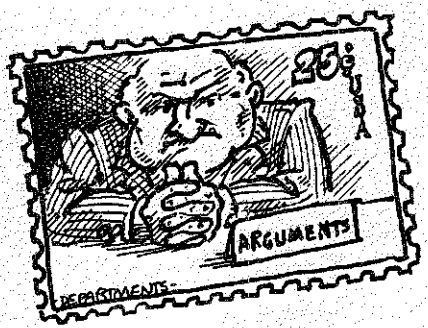


Lasertalk PC

A PostScript language development environment which provides interactive access to the PostScript interpreter, a PostScript language debugger/editor, onscreen previewer, dictionary browser and access to System 7 operation descriptions, online. Requires PC/AT or greater and a PostScript printer. Includes Microsoft Windows runtime. Suggested Retail Price $349. For more information, see your dealer or contact Emerald City Software at (415) 324-8088.

EMERALD CITY SOFTWARE

Number 241 on the Reader Service Card



You Want Arguments?

I'm in my second year reading your generally excellent publication. Since I write only in assembly language, my point of view is obviously biased. I feel

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

that there is too little about assembly language in your pages, even though almost all listings feature it.

Michael Abrash's article in volume 7.4 asks for opinions on whether more high-performance assembly language stuff or graphics stuff would be on the wish list. Since the former can include the latter, the choice is obvious.

All high-level languages appear to us "assemblerphiles" to be libraries of macros and subroutines of assembly language and the devotion of page after page to C, Pascal, and BASIC (rather than developing ever faster machine language pieces with the attendant skill-enhancing discussions of assemblers) seems to be a move in reverse.

I would like to see more arguments about both style and content of assembly code and will start with a small nitpick of John Otken's piece. He recommends, "Do not comment every line of code." My rejoinder is "Why not? It can't hurt." His other recommendations seem fine.

My personal favorite is A86, because it has some convenient features: **pop ax,cx,cx**, local labels, and the absence of a lot of needless verbosity—**extrn**, **group**, **assume**, etc.—plus it's *fast* and lends itself to convenient library use (no link step).

Bill Loughborough
Goldendale, Washington

The Request Line Is Open

I enjoyed your volume 7.4 article on PC communications. The serial-chip history was illuminating. I would like to encourage you to address PC parallel ports similarly. If one simply uses DOS services, they are fairly straightforward, but I have found no thorough references to working with them at the IO port and bit level (and few at any level). The *Programmer's Problem Solver* is, in fact, the only book I have found that approaches the subject. I even saw a comment in another book on writing PC device drivers that said, in effect, that "programming the parallel interface port is beyond the scope of this book"!

What particularly puzzles me is that I have some experience with parallel printer device handlers (drivers) on Digital PDP-11s. In that environment (using interrupts), they are among the simplest

Continued on page 10

STAGE#1 -Action Diagram Editor

STAGE#1 is a full screen editor that automates the generation and editing of action diagrams, the most versatile diagramming method available today. Action diagrams can be used to define software structure from the top-most overview down to the lowest level pseudo code. No other diagramming method has that capability.

More than 70 commands provide the familiar word processor functions, plus a few unique features. For example:

CONVERTS ACTION DIAGRAMS;

```
*Print_Character_Set

   For line Ø to 7
      Put leading spaces in the output string
      Put the line number and ': ' in the output string
      Calculate offset (line * 32)

      If line is not Ø or 4

         For count equals Ø to 31
            Character value is count plus offset
            Add character and space to the output string
      /* Endfor */

   /* Else */
      /* Add 'Control characters, can't print them!' to the */
        /* output string */
   /* Endifelse */

   /* Display the output string */
   /* Add CR and LF to the output string */
   /* Print the output string */
/* Endfor */

/* Print a formfeed */
```

INTO COMMENT LINES!

STAGE#1 also doubles as a normal text editor for entering program code. Direct access to DOS allows trial compilation or assembly without exiting the editor.

System requirements: IBM PC/XT/AT, PS/2 or compatible, PC or MS-DOS 2.1 or higher, minimum of 256KB of memory, CGA, EGA, VGA or monochrome monitor, two disk drives, printer with either the full ASCII or the IBM graphics (line-draw) character set. A plotter is not required.

INTRODUCTORY PRICE $159.00

~~Price: $180.00~~ (US), includes shipping (UPS ground) within the continental US. Washington residents add 7.8% sales tax. Educational discounts and site licensing are available. NOT copy protected. Specify 3 1/2 or 5 1/4 inch disk.

Why settle for a program that will produce only flowcharts and organization charts when you can have much more?

St Helens Software

SIMPLY POWERFUL

VISA MasterCard

Mt. St. Helens Software
P.O. Box 3319
Pasco, Washington 99302-3319

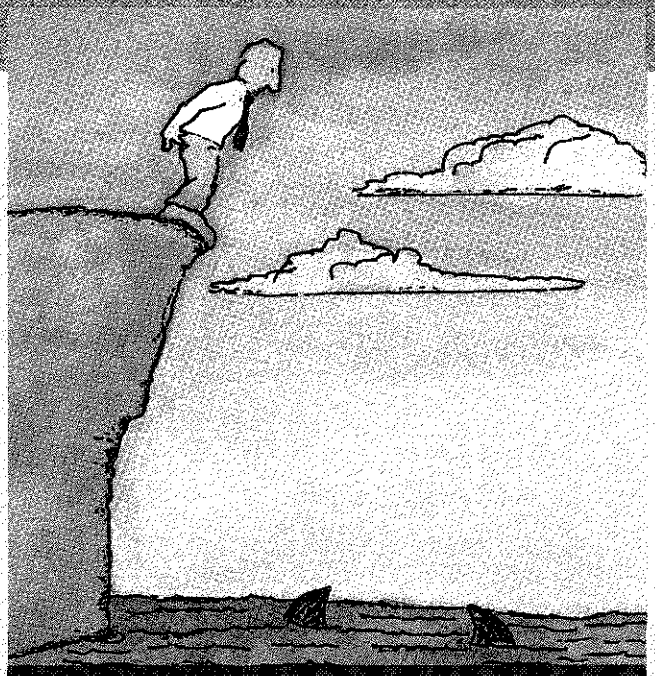
For orders or information:
(509) 547-2582
8 AM - 5 PM (PST)

St. Helens Software, Simply Powerful and STAGE#1 are trademarks of Mt. St. Helens Software. Other brand and product names are trademarks of their respective holders.

© Copyright 1989 Mt. St. Helens Software

Number **319** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Look Before You Leap.

You Can Have A Flexible, Powerful, Affordable CASE Tool.

Words of Caution and Encouragement for Those Choosing a CASE Tool

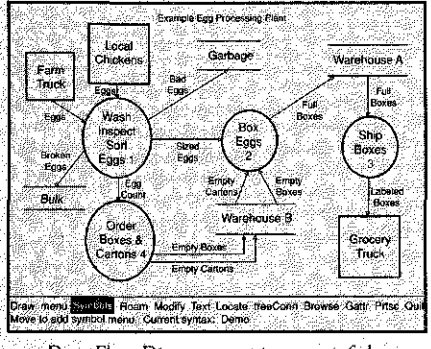
Your software projects are growing more complex, and maintenance is growing more expensive. You think that CASE tools could help, but some say CASE is more trouble than it's worth. What should you believe? Should you jump in? Be cautious. Expensive CASE tools with rigid methodologies demand a commitment of time and money so large that anything short of revolutionary success can seriously damage your budget and your career. Now, however, you can eliminate methodology grid-lock with an affordable, flexible, world-class CASE tool. Literally every programmer and analyst in your company can have the power of vsDesigner™ on their PS/2 or PC.

The Price/Performance Leader

vsDesigner costs only **$495** for a single-user license, but don't be fooled by its low price — vsDesigner has the features and power of CASE tools costing over ten times more. The Expert version of vsDesigner costs $1,995 for a single-user license and includes all vsDesigner features plus LAN support, design database reporting and the ability to modify or create methodologies.

Use Multiple Methodologies

vsDesigner supports the structured design methods you prefer: Yourdon, Gane-Sarson, Warnier-Orr, Ward-Mellor Real-Time, Entity-Relationship, Flow Chart, etc. It supports multiple methodologies or "syntaxes" simultaneously so that a design can contain layers (nodes) using different syntaxes. For example, you can use Data Flow Diagrams for the top level of your design and then decompose each module into Structure Charts, State Diagrams, Entity-Relationship Diagrams, etc.



Data Flow Diagrams are just one of the standard methodologies supported

Customize Methodologies Or Create Your Own

With the Expert version of vsDesigner, you can modify the syntaxes we supply, or you can even create your own syntaxes for ultimate flexibility. We believe that the tool should adapt to and then enforce your team's standards and practices rather than vice versa.

Quality Code From Quality Design

Although many CASE tool users have had significant productivity gains, structured design improves quality even more than productivity. The design checking programs included with vsDesigner allow you to check the logical integrity of your designs before you begin writing code. The resulting code will be more reliable, and maintenance costs will be much lower because operations are logically consistent.

Collaborate over a LAN

The Expert version of vsDesigner also adds LAN support so that all your designers can simultaneously work on the same design. Node locking allows design collaboration while preventing conflicting changes to the shared design.

The Expert version of vsDesigner also provides the database and reporting capabilities necessary to predict lines of code, number of modules, completion dates, responsibilities and project complexity.

One Picture Is Worth A Thousand Meetings

vsDesigner enhances the communication among team members by giving managers, team leaders, programmers and analysts a visual picture of your system's architecture making it faster and easier to communicate design changes throughout the team.

Fast Learning Curve

vsDesigner has powerful on-line help, and the User's Manual has an extensive tutorial. Your choice of Lotus-style menus or pop-up menus make command selection easy. The built-in word processor even emulates popular word processors to reduce learning time. Color graphics make your designs easier to understand.

A World-Class Design/Analysis Tool From The Best-Rated CASE Vendor

When you choose vsDesigner, you become a valued customer of Sage Software, the "Best-Rated CASE Vendor" in the Donohue 1989 Fortune 1000 Software Industry Report. Sage is committed to meeting your needs with excellent customer support, training and best-in-class tools including APS/PC and APS/MVS Application Generators, PVCS, PolyMake and vsDesigner.

Both versions of vsDesigner require an IBM-compatible PC or PS/2 with 640K RAM, EGA monitor and a hard disk. A mouse is recommended. Printers supported are: Epson FX85, FX286 & MX80; HP LaserJet+ and LaserJet Series II.

30-Day Money-Back Guarantee

To Order 1-800-547-4000

Outside North America call (301) 230-3307. Send checks and P.O.s to SAGE/POLYTRON, 1700 NW 167th Place, Beaverton, OR 97006. FAX: (503) 645-4576. Prices are N. American only.

SAGE SOFTWARE

Number **26** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Write Microsoft® C Programs In Half The Time

Instant-C™ means faster development and fewer bugs

Since 1984, *Instant-C* has been famous for two things: the fastest compile/link cycle in the industry, and a knack for finding bugs in already "debugged" code. Now, we're pleased to announce *Instant-C* 4.0, a bulletproof protected mode environment for developing Microsoft C applications.

Find bugs other compilers miss

Some classes of problems can cost you days or weeks to track down in a conventional environment. *Instant-C* catches them immediately, before the offending code is executed:

Dereferencing a null pointer
Stack overflow
Array reference out of bounds
Use of an uninitialized pointer
Overwriting a return address
Overwriting a code segment
Executing data

Instant-C catches parameter mismatches, even between modules, and function usage inconsistencies. Full checking is always active, even if you don't write prototypes—there are no special options to set, no separate `lint` program to run, no extra steps to take.

Fast compilation, fast execution

Instant-C is an in-memory incremental compiler. When you make a change to your program, *Instant-C* recompiles just the change. Imagine testing a 100,000 line program less than a second after changing it—that's *Instant-C*.

Instant-C has all the advantages of an interpreter without the drawbacks. Programs run under *Instant-C* at full speed, 50 to 500 times faster than interpreted programs.

Accelerate Microsoft C Development

Instant-C binds with your Microsoft C compiler's libraries to make an enhanced development environment. Your libraries become part of *Instant-C*, so you can call library functions interactively or from a program without ever re-linking.

Develop, test, and debug under *Instant-C*, then use Microsoft C to make the final version of your program.

Break the DOS 640K barrier

We know that just being Quick isn't enough. Under *Instant-C*, you can develop and debug programs that are too big for other environments—programs hundreds of thousands of lines long, in fact. Because *Instant-C* uses extended memory, your work isn't slowed down by swapping to disk.

> "The full speed of a real compiler, plus the added advantages of having the interpreter environment and debugger to work with. It's the best of both worlds for the experienced C programmer."—*PC Magazine*, Sept. 13, 1988

Build applications piece by piece

Omit modules you're not concerned with (you don't even need to write stub routines). Mix and match source and object code. Test as you go; since *Instant-C* is an incremental compiler, it only takes a second to try a new idea. You can go from specification to prototype to finished product without writing one line of throwaway code!

Popular function libraries supported

Instant-C is compatible with libraries from Blaise Computing, Copia, Faircom, MetaGraphics, Novell, Oakland Group, Raima, Roundhill, South Mountain Software, Vermont Creative Software, and others. Call us to ask about support not listed here.

Instant-C manages your code

If you've ever inherited a big program from someone else, you're familiar with the problems: a hard-to-read indenting style; confusingly named variables; data items of unknown type, or functions taking unknown parameters; not being able to find something. Not to mention side effects that creep in when you start rewriting.

Under *Instant-C*, you don't have to understand a whole project to work on part of it. Edit a function or data item without knowing where it's defined. Cross-reference a symbol, or browse through an outline of any module. View pop-up declarations and definitions at any time. Change a name globally, without worrying about scope or side effects. Have your code formatted to your specifications.

Instant symbolic debugging

You can use the full C language to describe when and where breakpoints should be triggered. At any breakpoint, you can execute any statement, inspect or change variables, and evaluate any expression (even those involving macros, locals and calls).

Instant-C lets you move around the program's execution stack with a single keystroke. You can edit any function, even an active one, recompile, and resume where you left off.

Instant-C supports dual-monitor debugging on all types of displays.

Proven technology

Since *Instant-C* debuted in 1984, it has doubled the productivity of many customers. Rational Systems' DOS extender, developed for *Instant-C* 4.0, was chosen by Lotus for 1-2-3® rel. 3.

Instant-C is a proven development environment. Prove it to yourself. Mention this ad when you order and we'll *double* our usual 30-day money back guarantee.

Instant-C is priced at $795.

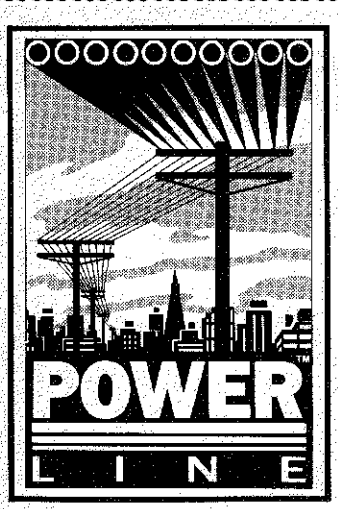
Rational Systems, Inc.
220 North Main St.
Natick, MA 01760
(508) 653-6006
Fax: (508) 655-2753

PJF1

Instant-C works with medium and large memory models, and requires DOS 2.0 or above, an 80286 or 80386 processor and at least 1M of memory. Microsoft is a registered trademark of Microsoft Corp. 1-2-3 is a registered trademark of Lotus Development Corporation.

Number **20** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Shove These In Your PC!

Pop POWERLINE II cards into your PC and drive them with our COMMANDO software. Get the results you want from your VOICE PROCESSING APPLICATIONS.

- **Voice Messaging**
- **Call Processing**
- **Audiotex**
- **Telemarketing**
- **Order Processing**
- **Call Distribution**

Our POWERLINE II multiple-port board functions as the heart of a Voice Communications System based on the IBM PC/XT/AT/386 architecture. Each card provides dedicated hardware support for voice processing, call analysis, touch-tone & rotary interaction, and telephone/PBX/Centrex integration.

COMMANDO's extensive function library allows you to rapidly generate sophisticated inbound and/or outbound applications supporting as many as 16 lines—even in the background!

And, all Talking Technology products carry a 2 YEAR WARRANTY. We design them to last!

POWERLINE II $599.00
COMMANDO $139.00
SINGLE LINE from . . . $295.00
VISA/MC/AMEX/COD

FOR ADDITIONAL INFORMATION OR SALES CALL:
(415) 652-9600
FAX (415) 652-5311

TALKING TECHNOLOGY, INC.
4383 PIEDMONT AVE., OAKLAND, CA 94611

Number 297 on the Reader Service Card

Playing with Matches

I was playing around with Adrian Crum's "Fast String Searching" code in *PJ* 7.4 and found a serious bug. I am not really familiar with the Boyer–Moore algorithm, but it seems to me that if the last character does not match, one needs to find the *rightmost* match for the end character, not the first. If the rightmost character occurs more than once in the target string, the search routine can fail.

For example, try finding the string "ABCBD" in the target "XABCBDX". The routine as published will get the second *B* from the target and scan for it in the search string. It will find it in position 2 and move the target pointer *past* the beginning of the matching string. After this happens, no match will be found. A better approach is to start from the right side of the search string and add the offset from the right side to the rightmost occurrence.

One other minor problem crops up with a classic 8086 "gotcha": the use of the JCXZ instruction after a REPNE CMPSB instruction. If the last character matches, CX will still be 0. I prefer using the JE/JNE after a REPNE, which will indicate if any match was found, including the last character.

I found this not by inspection, but by accident. A string that I knew was in the target was not being found, and it took a bit of head scratching on my part to find the problem. I have changed the routine to fix these problems:

```
        .
        .
        .
shift:
        push    di
        mov     al,[bx+si]
        add     di,bx           ;* point to end
        std                     ;* find RIGHTMOST match
        mov     bx,cx
        repne   scasb
        cld                     ;*
        jne     shift1          ;* test ZF not CX
        not     cx              ;* make count negative - 1
        add     cx,bx           ;* combine with len
        jne     ss_cx_ok        ;* make sure CX is not 0
        inc     cx              ;*  or SI won't change!
cx_ok:
        add     si,cx
        jmp short shift2
shift1:
        .
        .
        .
```

Mark E. Huss
W. Collingswood, New Jersey

of devices with the simplest of programming interfaces, whereas on the PC each different parallel printer seems to require slightly different timing.

I would certainly appreciate a survey on programming the parallel port—particularly trying to make the AT parallel port work in interrupt mode, which I have not yet accomplished.

Stephen Brenner
Nashville, Tennessee

Spellbinding Reading

Recently I was given two of your publications by a friend and was so impressed that I subscribed instantly. I found that once I started reading the magazines (volumes 7.3 and 7.4), I couldn't put them down. I have been looking for such a publication for two years. I'm a backyard programmer, and I was looking for a publication that would enlighten me in the latest techniques and that I could use as a training aid. Well, golly gee whiz, *Programmer's Journal* has all that I am looking for at this time.

I purchased MASM 5.1 and would like to see articles relating to assembly language programming, especially coding techniques and coding instructions to the assembler. I found the articles in volumes 7.3 ("Neural Nets") and 7.4

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

("Measuring Performance," "PC Communications and Protocols," "A Line of Code," and "A Millisecond Timer for BASIC") to be most interesting and educational.

Inform Michael Abrash that I want more assembly language topics.

W. R. Halm
Cincinnati, Ohio

Quick Disks

I recently purchased *Programmer's Journal* program disk 7.3 and 7.4, specifically for the BASIC code for the "Neural Nets" article in volume 7.3. As you mentioned in your README7.4 file on your 7.4 program disk, please send me Mr. Hardenbergh's code as soon as you receive it.

I have also noticed that you send *PJ* program disks promptly, usually within one week. This is a welcome change from other magazines, specifically *BYTE* and *Ciarcia's Circuit Cellar*.

I have been a casual newsstand reader of *Programmer's Journal* for two years and am impressed by the timeliness and value of your articles and code. Please find a check for $19.95 enclosed for a one-year subscription to *Programmer's Journal*. I am looking forward to receiving your magazine on a regular basis.

Gary R. Lambert
Chula Vista, California

Enquiring Minds

Do you know of a BASIC subroutine that allows a previously entered variable to be edited instead of requiring the user to reinput the entire string? If you have already published such a routine, please let me know, and I will order it.

Timothy M. Maloney
Knoxville, Tennessee

ETHAN WINER RESPONDS: Input routines are indeed an important part of many programs. Unfortunately, a good one requires more code than most magazines have room to print. The few that I have seen published are either very limited or not intended for use with BASIC.

Unlike BASIC's built-in INPUT statement, a routine designed for entering or editing strings must accept keystrokes one by one and handle each individually. A typical editing subprogram designed for Microsoft QuickBASIC will receive the string to be edited as an incoming parameter. The first step would be to print the existing string, position the cursor under the leftmost character, and then enter a DO loop that uses INKEY$ to wait for a keypress.

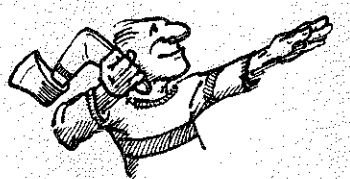
A SELECT/CASE block could then be used to handle each key, based on the action that is needed. For example, a character key will be added to the string at the current cursor position, while the backspace key would erase the previous character. Of course, there are many, many details involved in writing an input routine, such as restricting the input length, scrolling the string on the screen when characters are inserted or deleted, and so forth.

Another option you might consider is to purchase a commercial add-on toolbox package. Most include one or more input routines, and the better ones include source code, which lets you customize their operation. Further, studying the source code is an excellent way to learn how such routines are written.

Correction

[In the last issue, Notes from the Editor mentioned the release of Microsoft FORTRAN 5.0 for DOS. We neglected to mention that this version generates code and works under both DOS 3.0 and OS/2 version 1.1 or higher. —Ed.]

We at *Programmer's Journal* like to hear from our readers. Please address correspondence to Managing Editor, *Programmer's Journal*, 150 N. 4th St., Springfield, OR 97477. You can also contact us via fax (503-746-0071), CompuServe (user ID 76676,427), or MCI Mail (user ID 336-4248). All letters become the property of *Programmer's Journal* and may be edited for publication.



...where AX is a return value...

T-SHIRTS

The ***Programmer's Journal*** issue special for Volume 7.6 is **T-Shirts!** Normally $12.00, any of our very fine, high-quality T-shirts can for a limited time be obtained for only **$10.00!** This reduced price still includes postage and handling! **Ordering is easy** with your Visa or Mastercard and our toll-free number:

800-234-0386

See our ad on page 96 for details.

What C was *supposed* to be.

P.D.Q.™

P.D.Q. is the closest you'll come to a pure assembler program, without actually having to write in assembly language...

Introducing **P.D.Q.** This remarkable library lets you write programs that are extremely small and fast, while using a high- level language you already know. **P.D.Q.** can produce a "hello world" program in Microsoft® QuickBASIC with a total .EXE file size of only 850 bytes. Imagine a DOS "filter" program taking 1400 bytes of memory. Or a TSR printer set-up utility in only 3K. That's right, TSR and interrupt handling are built into the **P.D.Q.** library!

P.D.Q. is a replacement for the BCOM libraries that come with QuickBASIC and BASIC 6.0. It's ideal for writing small to medium applications and TSR utilities, where program size and execution speed is critical.

Why struggle with C when you can create even smaller and faster programs with a language you already know.

* Includes Numerous Example Programs
* Full Source Code
* Free Demo Disk
* **P.D.Q.** is only $99

Seeing is believing. Please write or call:

Crescent Software Inc.

11 Grandview Avenue,
Stamford, CT 06905
(203) 846-2500

Please add $6 for shipping.
Visa, MasterCard, COD accepted

Number **83** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Wealth Through Project Growth

by William H. Roetzheim

Many consultants I talk to are under the impression that revenue growth is simply a matter of effectively meeting customers' needs. In theory, as they prove themselves they will be rewarded with larger and larger contracts. Let's bypass this myth and put your company on the fast track to success. The trick is really quite obvious: Strive to expand every contract you work on into a large contract.

Acorns into Oak Trees

Each contract, no matter how small or limited in scope, is a seed from which a large contract might sprout. Never lose sight of the fact that projects are also marketing opportunities. Let me give you a couple of examples.

I was asked to prepare and present a two-week training class to a research laboratory on a version of the UNIX operating system. During presentation of the class, I took advantage of the close interaction between me and the students to discover what UNIX-related work was going on and what areas were giving the lab problems. After the class was over, I was able to talk to appropriate program managers and suggest ways our company could assist them in their work. That small project to perform training resulted in follow-up work worth well over ten times the value of the original contract.

On another occasion, I was asked to develop custom interface hardware and software to allow a set of integrated tactical decision-aid software programs to automatically gather required information from shipboard sensors. This project gave me an opportunity to become familiar with the existing software and to demonstrate my software-development capabilities to the customer. This relatively small contract allowed me to win a subsequent contract to modify and enhance the tactical-decision–aid software. The value of the follow-up work was over two million dollars.

The lesson here is simple: Every project, no matter how small or mundane, should be considered a marketing opportunity that might lead to work that is more interesting, more profitable, or simply greater in scope.

Project Growth Strategies

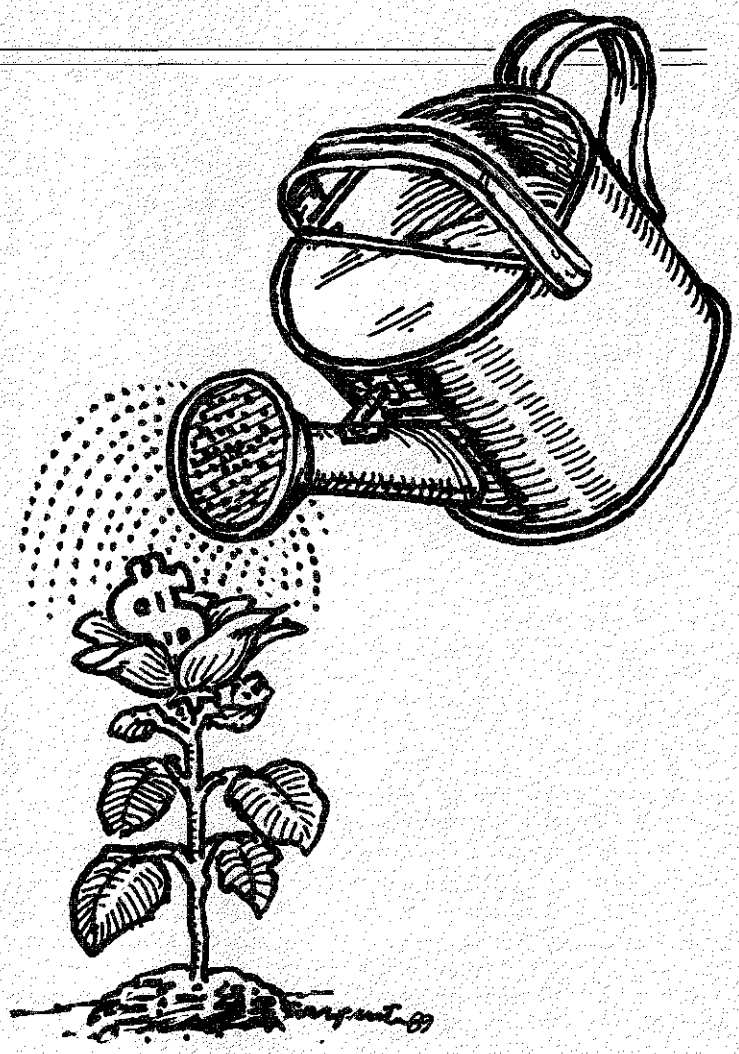
There are three common approaches to expanding a project: Expand the current work, obtain related work, and obtain unrelated work with the same customer.

Expanding the Scope

One of the simplest methods of augmenting a project is simply to expand the scope of the current project. This is normally the easiest option from a contractual standpoint, requiring only a modification to the existing contract. This advantage is especially important in government contracting, where new work will often require competitive procurement. Here are some approaches to expanding the scope of a project:

- As the project progresses and you obtain a better understanding of the application and the current software solution, many possible improvements to the requirements or the design will normally become obvious. These improvements can be proposed to the customer as engineering changes or as suggested follow-up enhancements, both of which will normally involve additional funds for your company.
- Prepare a detailed proposal to provide software maintenance.
- Most development programs do not include adequate planning for verification and validation of the operational value of the program. Propose a quantitative study to validate the effectiveness of the program.
- Training is an area that is often not considered by the customer until the last minute. Recommend a training curriculum for the end users of the program.
- Many contracts do not require sufficient documentation, offering an opportunity to propose contract modifications for additional programmer or user documentation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



A Foot in the Door

Every project should be considered a foot in the door toward additional related work. Although related work will normally require a new contract and possible competition, you bring several valuable advantages to the competition. Your experience with the current project will often make you uniquely qualified to perform related work. The current project may have included the development of sample data, test equipment or software, or other development tools. You may be able to use your access to these items to your advantage if they will apply to the new work.

On government contracts, many companies develop key tools such as these at their own expense to avoid requirements to turn the work over to the government at the end of the contract. If the contract involved the use of an uncommon development language, operating system, database management system, or such, your company's experience in this area gives you an edge over your competitors. This is one reason nonstandard languages and operating systems are sometimes recommended for a given project. Working with unusual hardware provides a similar competitive advantage. Finally, if the project requires some specialized government clearances, you will have a tremendous advantage over your competitors, especially if they cannot even apply for the specialized clearance without a project in hand to justify a "need to know."

Personal Rapport

Finally, your personal rapport with the customer, combined with demonstrated superior performance on the project, will often allow you to locate and obtain unrelated work from the same (or a related) customer.

Your Role as Consultant

Your role as a consultant in marketing for growth involves:

- Identifying opportunities for expanding the scope of the current work, obtaining related work, or obtaining unrelated work with the same customer.
- Managing your project in a manner that will build a positive image of your company in the customer's mind, ensuring his or her confidence in your ability to perform the additional work.
- Designing systems, whenever possible and ethical, in a manner that will give your company a competitive edge over other companies when soliciting follow-up work.
- Making specific proposals to the customer for additional work.
- Assisting the customer as required with cost estimates, advice about contract procedures, and preparation of rough delivery orders or statements of work.

In a future issue, we will learn about the five market segments available to computer consultants and discover how to penetrate each of these segments.

This article is based, with permission, on the author's book Structured Computer Project Management *(Prentice-Hall, 1988).*

William H. Roetzheim received his M.B.A. in 1983 and is currently an M.S. candidate in computer science. He has worked as senior project manager for Tetra Tech Services (a Honeywell subsidiary) and is the author of several books, including Structured Computer Project Management *(Prentice-Hall, 1988) and* Proposal Writing for the Data Processing Consultant *(Prentice-Hall, 1986). Mr. Roetzheim has also written Structured Project Manager's Toolbox (SPMT), an IBM-PC–based project-management tool that is tailored to software-development projects. Mr. Roetzheim may be contacted at 3891 American Avenue, La Mesa, CA 92041, 619-464-0182.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VGA and EGA Split Screens

by Michael Abrash

Installment 15: In which we learn our way around the nifty but surprisingly tricky split screen of the VGA and EGA.

The ability to split the screen into two largely independent portions—one displayed above the other on the screen—is one of the more intriguing capabilities of the VGA and EGA. The split-screen feature can be used for pop-ups (including pop-ups that slide smoothly onto the screen) or simply to display two separate portions of display memory on a single screen. While it's possible to accomplish the same effects without using the split screen, software solutions tend to be hard to implement and slow.

In contrast, the basic operation of the split screen is simple. All is not tea and crumpets, however: For one thing, there are a few tricks to programming the split screen. For another, the EGA's split-screen implementation is a little buggy. Finally, panning with the split screen enabled is not so simple as it might seem. All in all, we have some ground to cover. Let's start with the basic operation of the split screen.

How the Split Screen Works

The operation of the split screen is simplicity itself. A split-screen start-scan-line value is programmed into two EGA registers or three VGA registers (more on exactly which registers in a moment). The video circuitry begins to scan display memory for video data starting at the address specified by the start-address registers, just as it normally would. When the video circuitry encounters the specified split-screen start scan line in the course of scanning video data onto the screen, it completes that scan line normally and then resets to 0 the internal pointer that addresses the next byte of display memory to be read for video data. Display memory from address 0 onward is then scanned for video data in the usual way, progressing toward the high end of memory. At the end of the frame, the pointer to the next byte of display memory to scan is reloaded from the start-address registers, and the whole process starts over.

The net effect: The contents of display memory starting at offset 0 are displayed starting at the scan line following the specified split-screen start scan line, as shown in the figure on page 16. It's important to understand that the scan line that matches the split-screen scan line is not part of the split screen—the split screen starts on the *following* scan line. So, for example, if the split-screen scan line is set to 0, the split screen actually starts at scan line 1, the second scan line from the top of the screen.

If both the start address and the split-screen start scan line are set to 0, the data at offset 0 in display memory is displayed as both the first scan line on the screen *and* the second scan line. There is no way to make the split screen cover the entire screen—it always comes up at least one scan line short.

So, where is the split-screen start scan line stored? The answer varies a bit, depending on whether you're talking about the EGA or the VGA. On the EGA, the split-screen start scan line is a nine-bit value, with bits 7–0 stored in the Line Compare register (CRTC register 18h) and bit 8 stored in bit 4 of the Overflow register (CRTC register 7). Other bits in the Overflow register serve as the high bits of other values, such as the vertical total and the vertical blanking start. Since EGA registers are—alas!—not readable, you must

It's important to understand that the scan line that matches the split-screen scan line is not part of the split screen—the split screen starts on the following scan line.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

know the correct settings for the other bits in the Overflow registers to use the split screen on an EGA. Fortunately, there are only two standard Overflow-register settings on the EGA: 11h for 200-scan-line modes and 1Fh for 350-scan-line modes.

The VGA, of course, presents no such problem in setting the split-screen start scan line, for it has readable registers. However, the VGA supports a ten-bit split-screen start-scan-line value, with bits 8–0 stored just as with the EGA and bit 9 stored in bit 6 of the Maximum Scan Line register (CRTC register 9).

Turning the split screen on involves nothing more than setting all bits of the split-screen start scan line to the scan line after which you want the split screen to start appearing. (Of course, you'll probably want to change the start address before using the split screen; otherwise, you'll just end up displaying the memory at offset 0 twice—once in the normal screen and once in the split screen.) Turning off the split screen is a simple matter of setting the split-screen start scan line to a value equal to or greater than the last scan line displayed; the safest such approach is to set all bits of the split-screen start scan line to 1. (That is, in fact, the split-screen start-scan-line value programmed by the BIOS during a mode set.)

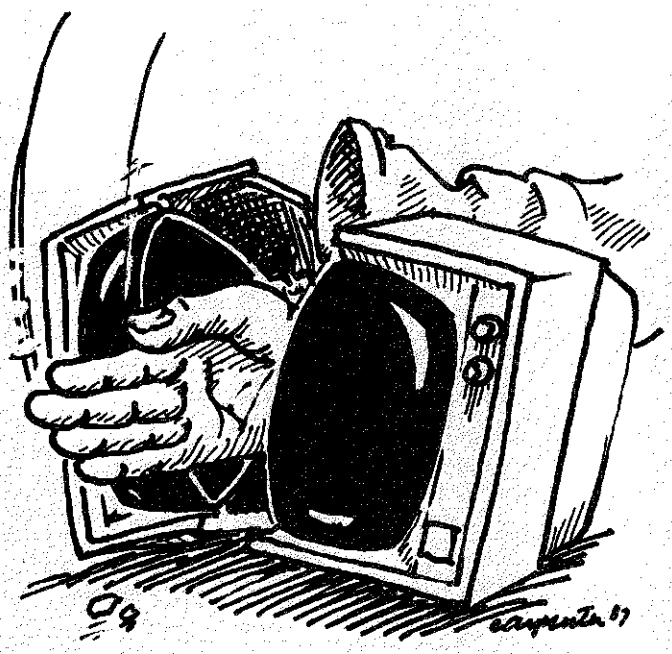
The Split Screen in Action

All of the above points are illustrated by Listing 1. The listing fills display memory starting at offset 0 (the split-screen area of memory) with text identifying the split screen, fills display memory starting at offset 8000h with a graphics pattern, and sets the start address to 8000h. At this point, the normal screen is displayed (the split-screen start scan line is still set to the BIOS default setting, with all bits equal to 1, and so the split screen is off), with the pixels based on the contents of display memory at offset 8000h. The contents of display memory between offset 0 and offset 7FFFh are not visible at all.

Listing 1 then slides the split screen up from the bottom of the screen, one scan line at a time. The split screen slides halfway up the screen, bounces down a quarter of the screen, advances another half-screen, drops another quarter-screen, and finally slides all the way up to the top. If you've never seen the split screen in action, you should run Listing 1—the smooth overlapping of the split screen on top of the normal display is a striking effect.

Listing 1 isn't done just yet, however. After a keypress, the listing demonstrates how to turn the split screen off (by setting all bits of the split-screen start scan line to 1). After another keypress, Listing 1 shows—by setting the start address to 0 and then flipping back and forth between the normal screen and the split screen with a split-screen start scan line setting of 0—that the split screen can never cover the whole screen. Both the normal screen and the split screen display the same text, but the split screen displays it one scan line lower, because the split screen doesn't start until *after* the first scan line, and that produces a jittering effect as the program switches the split screen on and off. (On the EGA, the split screen may display *two* scan lines lower, for reasons I'll discuss shortly.)

Finally, after another keypress, Listing 1 ends.



VGA and EGA Don't Mix

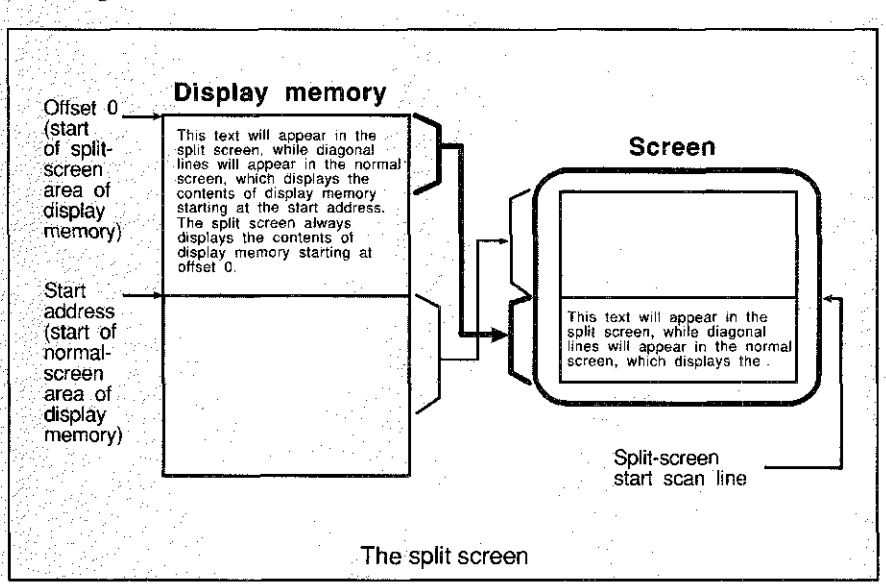
You must correctly set the IS_VGA equate at the start of Listing 1 for the adapter the code will run on to ensure that the program will perform properly. This equate determines how the upper bits of the split-screen

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

start scan line are set by **SetSplitScreenRow**. If IS_VGA is 0 (specifying EGA), then bit 8 of the split-screen start scan line is set by programming the entire Overflow register to 1Fh; this is hard-wired for the 350-scan-line modes of the EGA. If IS_VGA is 1 (specifying VGA), then bits 8 and 9 of the split-screen start scan line are set by reading the registers they reside in, changing only the split-screen-related bits, and writing the modified settings back to their respective registers.

The VGA version of Listing 1 won't work on an EGA, because EGA registers aren't readable. The EGA version of Listing 1 won't work on a VGA, because VGA monitors require different vertical settings than EGA monitors and because the EGA version doesn't set bit 9 of the split-screen start scan line. In short, there is no way that I know of to support both VGA and EGA split screens with common code; separate drivers are required.

By the way, Listing 1 operates in mode 10h because that's the highest-resolution mode the VGA and EGA share. That's not the only mode the split screen works in, though; in fact, it works in all modes, as we'll see later.


Display memory
Offset 0 (start of split-screen area of display memory)
This text will appear in the split screen, while diagonal lines will appear in the normal screen, which displays the contents of display memory starting at the start address. The split screen always displays the contents of display memory starting at offset 0.
Start address (start of normal-screen area of display memory)
Screen
This text will appear in the split screen, while diagonal lines will appear in the normal screen, which displays the
Split-screen start scan line
The split screen

Setting Split-Screen Registers

Setting the split-screen-related registers is not so simple a matter as merely outputting the right values to the right registers; timing is also important. The split-screen start-scan-line value is checked against the number of each scan line as that scan line is displayed, which means that the split-screen start scan line potentially takes effect the moment it is set. In other words, if the screen is displaying scan line 15 and you set the split-screen start to 16, that change will be picked up immediately and the split screen will start after the next scan line. This is markedly different from changes to the start address, which take effect only at the start of the next frame.

The instantly effective nature of the split screen is a bit of a problem, not because the changed screen appears as soon as the new split-screen start scan line is set—that seems to me to be an advantage—but because the changed screen can appear *before* the new split-screen start scan line is set. Remember, the split-screen start scan line is spread out over two or three registers; what if the incompletely changed value matches the current scan line after you've set one register but before you've set the rest? For one frame, you'll see the split screen in the wrong place—possibly a *very* wrong place—resulting in jumping and flicker.

The solution is simple: Set the split-screen start scan line at a time when it can't possibly match the currently displayed scan line. The easy way to do this is to set it when there isn't any currently displayed scan line—that is, during vertical nondisplay time. One safe time that's easy to find is the start of the vertical-sync pulse, which is typically pretty near the middle of vertical nondisplay time, and that's the approach I've followed in Listing 1. I've also disabled interrupts during the period when the split-screen registers are being set. This isn't absolutely necessary, but if it's not done, an interrupt may occur between register sets and delay the later register sets until display time, again causing flicker.

One interesting result of setting the split-screen registers at the start of vertical sync is that it has the effect of synchronizing the program to the display adapter's frame rate. No matter how fast a computer Listing 1 runs on, the split screen will move at a maximum rate of once per frame. This is handy for regulating execution speed over a wide variety of hardware performance ranges; however, be aware that the VGA supports 70-Hz frame rates in all non-480-scan-line modes, while the VGA in 480-scan-line-modes and the EGA in all color modes support 60-Hz frame rates.

The EGA Split-Screen Problem

I mentioned earlier that the EGA's split screen is a little buggy. "How?" you may well ask, particularly given that Listing 1 illustrates that the EGA split screen is pretty much functional.

The bug is this: The first scan line of the split screen—the scan line starting at offset 0 in display memory—is displayed not once but twice. In other words, the first line of split-screen display memory—and only the first line—is replicated an extra time, pushing all the other lines down one.

That's not a fatal bug, of course; in fact, if the first few scan lines are identical, it's not even noticeable. The EGA's split-screen bug can produce visible distortion given certain patterns, however; so you should try to make the top few lines identical when designing split-screen images that might be displayed on EGAs. You should check how your split-screens look on both VGAs and EGAs in any case.

Don't count on the EGA's split-screen bug; that is, don't rely on the first scan line being doubled and build that effect into your split screens. There's no guarantee that all EGA clones copy this bug, and it's a certainty that the VGA didn't copy it.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

There's another respect in which the EGA is not so good as the VGA when it comes to the split screen, and that's in the area of panning when the split screen is on. This isn't a bug—it's just one of the many areas in which the VGA's designers learned from the shortcomings of the EGA and went the EGA one better.

Split Screen and Panning

Way back in the first installment of "On Graphics" (*PJ* 5.1, January/February 1987), I presented a program that performed horizontal smooth panning. Horizontal smooth panning consists of two parts: byte-by-byte (8-pixel) panning by changing the start address, and pixel-by-pixel intrabyte panning by setting the Pel Panning register (AC register 13h) to adjust alignment by 0 to 7 pixels. (IBM uses the word *pel* instead of *pixel* in much of their documentation; hence *pel panning*.)

Horizontal smooth panning works just fine, although I've always harbored some doubts that any one horizontal-smooth-panning approach works properly on all clones (more on this later). There's a catch when using horizontal smooth panning with the split screen up, though, and it's a serious catch: You can't byte-pan the split screen (which always starts at offset 0, no matter what the setting of the start-address registers)—but you *can* pel-pan the split screen.

To put it another way, when the normal portion of the screen is horizontally smooth-panned, the split-screen portion moves a pixel at a time until it's time to move to the next byte, when it jumps back to the start of the current byte. As the top part of the screen moves smoothly about, the split screen will move and jump, move and jump, over and over.

Believe me, it's not a pretty sight.

What's to be done? On the EGA, nothing. Unless you're willing to have your users' eyes doing the jitterbug, don't use horizontal smooth scrolling when the split screen is up. Byte panning is fine—just don't change the Pel Panning register from its default setting.

On the VGA, there is recourse: A VGA-only bit, bit 5 of the AC Mode Control register (AC register 10h), turns off pel panning in the split screen. When this bit is set to 1, pel panning is reset to 0 before the first line of the split screen and remains 0 until the end of the frame. This doesn't allow you to pan the split screen horizontally, mind you—there's no way to do that—but it does let you pan the normal screen while the split screen stays rock solid. This can be used to produce an attractive "streaming tape" effect in the normal screen while the split screen is used to display nonmoving information.

An Illustrative Example

Listing 2 illustrates the interaction of horizontal smooth panning with the split screen as well as the suppression of pel panning in the split screen. *[NOTE: Because of space limitations, Listing 2 is not printed in this issue. It is available on the Code Disk for this issue as well as through our bulletin board services (see pages 54 and 96). —Ed.]* The listing creates a virtual screen 1024 pixels across by setting the Offset register (CRTC register 13h) to 64, sets the normal screen to scan video data starting far enough up in display memory to leave room for the split screen starting at offset 0, turns on the split screen, and fills in the normal screen and split screen with distinctive patterns.

Next, Listing 2 pans the normal screen horizontally without setting bit 5 of the AC Mode Control register to 1. As you'd expect, the split screen jerks about quite horribly. After a keypress, the listing sets bit 5 of the Mode Control register and pans the normal screen again. This time, the split screen doesn't budge an inch.

By the way, if IS_VGA is set to 0 in Listing 2, the program will assemble in a form that will run on the EGA (and only the EGA). Pel-panning suppression in the split screen won't work in this version, however, because the EGA lacks the capability to support that feature. When the EGA version runs, the split screen simply jerks back and forth during both panning sessions.

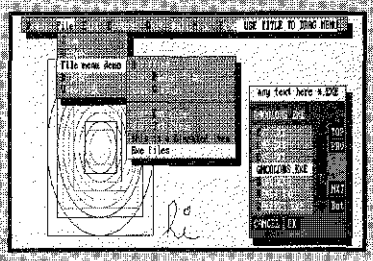
Setting and Reading Registers

There are a few interesting points regarding setting and reading registers to be made about Listing 2. First, bit 5 of the AC Index register

FINALLY...
a professional menu system
in a *graphics* environment for
TURBO C, MICROSOFT C,
TURBO PASCAL:™

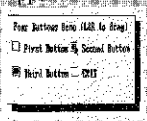
graphics-MENU

ONLY $99 (BGI) SOURCE AVAIL.

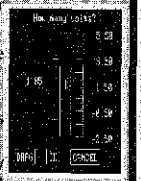


NOW AVAILABLE FOR:
Borland Graphics Interface

graphics-MENU from IS AND SYSTEMS is a comprehensive utilities package that allows the developer of graphics software applications to quickly create a user friendly interface. You can spend more time focusing on the internals of your application.



- Mouse handling.
- **Full** keyboard support.
- Pulldown Menu & Popup messages.
- Underlying graphics automatically saved.
- Mouse can drag menu/message to reveal image beneath.
- Shadowed menus create a 3-D effect.
- Disabled entries and divider lines.
- Vertical List optionally tiled.
- Horizontal list with corresponding pulldown menus that may have nested submenus to any depth.
- Popup message boxes, with or without user prompt.
- Button menus in any arrangement.
- Controls & gauges with analog style entry.
- Analog clock settable by moving its hands with the mouse.
- Limitless hybrids of the above menus for customization.
- Geometric interface functions: point rotation, distance, degrees/radians conversion, true 4-quadrant arc tangent.
- **Color Customizer** allows complete color control.
- **Data entry** module (optional add-on) allows easy forms creation with data validation and range checking.



We are also including two utility programs; CUREDIT and MenuDesigner. **CUREDIT** is a Cursor icon Editor that enables you to create custom cursor icons and associate them with any mouse button or chord.
MenuDesigner is a very powerful utility that enables you to create and view complex horizontal & vertical menu structures on-screen and then writes **TP, or TC/MS-C** code to implement this structure in your application program.

30 day money back guarantee on non-source

BGI VERSION
Requires: Borland Turbo Pascal or Turbo C, IBM with graphics, hard disk.
Note: Selected source includes menuing, messaging, geometric functions only.

for TP or TC	$ 99
selectede source- add	$149
MenuDesigner	$ 49
Data Entry Module	$ 59

META VERSION
Requires: MetaWindow (from Metagraphics Software), IBM with graphics, hard disk.

(pricing includes MenuDesigner)

for TP or TC or MS-C	$149
source(TP)- add	$149
source(C)- add	$199
Data Entry Module	$ 59

Island Systems
7 Mountain Rd. Burlington MA 01803
(617) 273-0421
FAX: (617) 933-1152

Number 317 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

should be set to 1 whenever palette RAM is not being set—which is to say all the time in your code, because palette RAM should normally be set via the BIOS. When bit 5 is 0, video data from display memory is no longer sent to palette RAM, and the screen becomes a solid color—not normally a desirable state of affairs.

Recall also that the AC Index and Data registers are both written to at I/O address 3C0h, with the toggle that determines which one is written to at any time switching state on every write to 3C0h. This toggle is reset to index mode by each read from the Input Status 0 register (3DAh in color modes, 3BAh in monochrome modes). The AC Index and Data registers can also be written to at 3C1h on the EGA, but not on the VGA; so steer clear of that practice.

On the VGA, reading AC registers is a bit different from writing to them. The AC register currently addressed by the AC Index register can be read from 3C1h; reading does not affect the state of the AC index/data toggle. Listing 2 illustrates reading from and writing to the AC Index and Data registers.

Finally, setting the start-address registers (CRTC registers 0Ch and 0Dh) has its complications. As with the split-screen registers, the start-address registers must be set together and without interruption at a time when there's no chance of a partial setting being used for a frame. However, it's a little more difficult to know when that might be the case with the start-address registers than it was with the split-screen registers, because it's not clear when the start address is used.

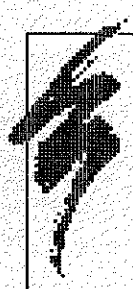
You see, the start address is loaded into the EGA's or VGA's internal display-memory pointer once per frame. The internal pointer is then advanced, byte by byte and line by line, until the end of the frame (with a possible resetting to 0 if the split-screen line is reached) and is then reloaded for the next frame. That's straightforward enough; the real question is, exactly when is the start address loaded?

Richard Wilton's excellent book *Programmer's Guide to PC & PS/2 Video Systems* (Microsoft Press) says (on page 391) that the start address is loaded at the start of the vertical-sync pulse. (Wilton calls it "vertical retrace," which can also be taken to mean vertical nondisplay time, but given that he's testing the vertical-sync status bit in the Input Status 0 register, I assume he means that the start address is loaded at the start of vertical sync.) Consequently, he waits until the *end* of the vertical-sync pulse to set the start-address registers, confident that the start address won't take effect until the next frame.

I'm sure Wilton is right when it comes to the real-McCoy IBM VGA and EGA, but I'm less confident that every clone out there loads the start address at the start of vertical sync. For that very reason, I generally advise people not to use horizontal smooth panning unless they can test their software on all the makes of display adapters it might run on. I've used Wilton's approach in listings 1 and 2, and so far it works fine; but be aware that there are potential—albeit unproven—hazards in relying on the setting of the start-address registers to occur at a specific time in the frame.

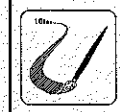
Incidentally, Wilton's book contains a useful overview of the various smooth-scrolling and smooth-panning features of the VGA and EGA, along with working code, although so far as I can tell it doesn't cover the VGA's split-screen panning suppression capabilities. The book also confirms my feeling that some EGA clones may not duplicate the first split-screen line.

The interaction of the start-address registers and the Pel Panning register is worthy of note. In Listing 2, after waiting for the end of vertical sync to set the start address, I wait for the start of the *next* vertical sync to set the Pel Panning register. That's because the start address doesn't take effect until the start of the next frame, but the pel-panning setting takes effect at the start of the next line; if we were to set the pel panning at the same time we set the start address, we'd get a whole frame with the old start address and the new pel-panning settings mixed together, causing the screen to jump.



GRAPHICS!

Add lightning fast graphics to your programs quickly and easily through the popular PCX file format. Why reinvent the wheel? Make your programs immediately compatible with hundreds of packages from Aldus PageMaker to ZSoft's PC Paintbrush with these linkable graphic libraries.



"An exceptional product" - Programmer's Journal, Aug

NEW! Version 3.5 of the **PCX Programmer's Toolkit** gives you over 60 powerful functions to manipulate bitmapped graphics. Use Virtual screens, Super VGA modes, LIM 4.0 support, a 300 page manual, 9 utilities including screen capture and display, and the fastest routines on the market. **$195**

Need Special Effects, but caught in a GRASP?

Why create a demo when you can create the real thing? Don't be trapped in a slideshow editor or demo program when you can use **PCX Effects** for the PCX Toolkit and your favorite programming language. A Music Language and spectacular effects for exploding your graphics! **$99**

Blazing Graphics Text

With **PCX Text** you can display text with graphics as fast as it always should have been. Display characters, strings, fixed and proportional text, background transparency, and more. Includes a font editor, 85 fonts, and text utilities for blazing graphics bitmapped text. **$149**

All packages support 12 compilers for C, Pascal, Basic, Fortran, Assembly, and Clipper. All modes of the Hercules, CGA, EGA, VGA, and Super VGA adapters are supported, up through 800x600x256 (22 modes in all). Assembly Language source code is optionally available. Trademarks are property of their respective holders.

G·E·N·U·S MICROPROGRAMMING

11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737

No Royalties! 30-day Money Back Guarantee.
VISA/MC/AMEX/COD/PO accepted.

1-800-227-0918

Number **261** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

As with the split-screen registers, it's safest to set the Pel Panning register during nondisplay time.

What if you wanted to pan faster? Well, you could of course just move two pixels at a time rather than one; I assure you no one will ever notice when you're panning at a rate of 10 or more times per second. Alternatively, you could double panning speed by loading the pel-panning setting for the current frame at the start of vertical sync and then loading the start-address setting for the next frame at the end of the same vertical sync. The start-address setting wouldn't take effect until the next frame, by which time you'll have loaded yet another pel-panning setting. This interleaving of pel-panning and start-address settings would allow you to pan as many as 60 times per seconds, or 70 on the VGA in 70-Hz frame-rate modes—the idea is that you could pan once per frame. (It should be obvious that you can't pan any faster than once per frame.)

Split Screen in Other Modes

So far, we've discussed the split screen only in mode 10h. What about in other modes? Generally, the split screen works in any mode. The basic rule is that when the scan line on the screen matches the split-screen scan line, the internal display-memory pointer is reset to 0. I've found this to be true even in oddball modes, such as line-doubled CGA modes and the 320-by-200 256-color mode (which is really a 320-by-400 mode with each line repeated). For split-screen purposes, the VGA and EGA seem to count purely in scan lines, not in rows or doubled scan lines or the like. However, I have run into small anomalies in those modes on clones, and I haven't tested all modes; so be careful when using the split screen in modes other than modes 0Dh–12h, and test your code on a variety of hardware.

(Come to think of it, I warn you about the hazards of running fancy EGA/VGA code on clones pretty often, don't I? Ah, well, just one of the hazards of the diversity and competition of the PC market! It is a fact of life, though: If you're a commercial developer and don't test your video code on at least half a dozen

NEW VERSION 4.5

DYNAMITE DEMOS

Proteus Prototype/Demo System

Why fizzle with an outdated demo program when you can explode with Proteus?

Proteus is the first prototype/demo system to combine complete simulation capability, a powerful Screen Painter, video effects, custom sound and much more in a program you don't have to be a genius to figure out.

And now Proteus supports all graphic modes for the CGA, EGA and VGA in the popular PCX file format.

"A joy to use." Computer Language

- Complete Simulation Capability
- Powerful Screen Painter
- Bit-Mapped Graphics
- Video Effects
- Custom Sound

Don't Let Your Next Demo be a Dud!

Find out why software developers from Ashton-Tate to ZyLogic have switched to Proteus.

1-800-634-9986

$199. No royalties. 30-day Money Back Guarantee. MC/VISA/COD/PO accepted.

Helios

P.O. BOX 22869, SEATTLE, WA 98122

Number 258 on the Reader Service Card

Would you like to be able to include a hypertext help system in your programs like the big guys: **Microsoft and Borland?***

Introducing... Help System II

We've enhanced the hypertext help system our hypertext products have used for 2 years so that you can include them in your programs:

- **Context Sensitivity**
- **Help for 84 different key combinations**
- **Requires less than 10K**
- **Data compression up to 80 percent**

$199.95 buys object files for Pascal, C, Basic and technical support

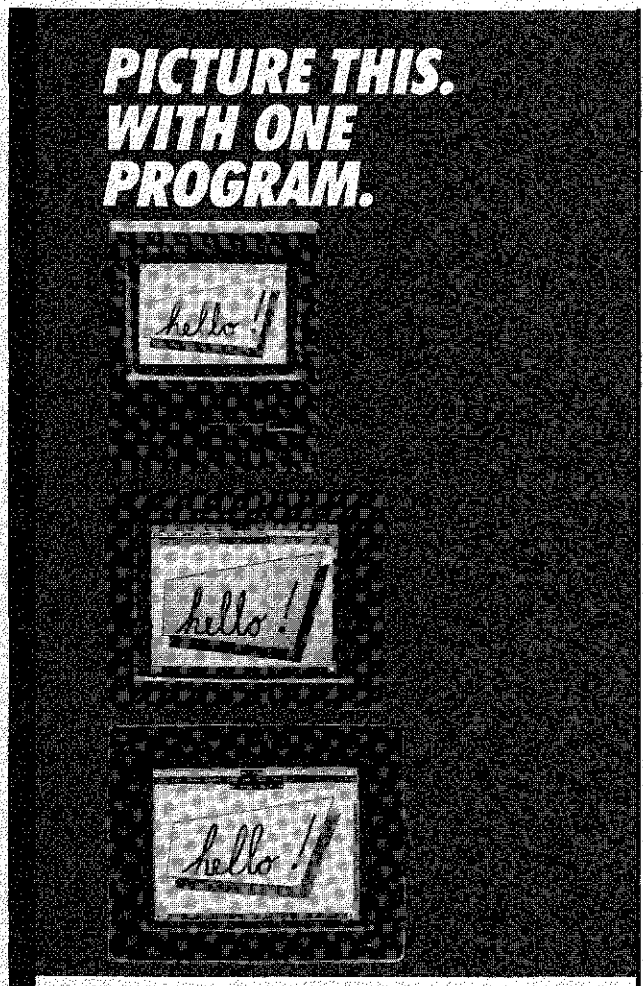
To order: **(203) 368 - 0632**

NTERGAID Inc.
955 Connecticut Ave.
Bridgeport, CT 06607

*** At least one of these companies purchased our product before they released theirs.**

Number 302 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Get a window on the whole personal computer world with XVT.™ An API that works with Microsoft Windows, Presentation Manager and the Macintosh.

XVT lets you write one set of source code for all three environments. And development is speeded-up by high-level XVT calls which generate multiple calls in the native environment.

WINDOWS ARE WINDOWS AND A MAC IS A MAC.

Because XVT generates native toolkit calls, each graphics environment retains the personality its users expect. XVT preserves the unique "look-and-feel" of each.

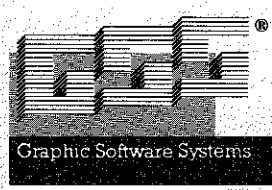
CALL (503) 641-2455.

ASK FOR DEPARTMENT PJ

IT'S A WINDOW OF OPPORTUNITY.

Cut your applications backlog down to size. Call GSS for more information, or a rapid response to your order for XVT. It's a product the whole personal computer world agrees on.

XVT is a trademark of API Ltd.
The GSS logo and GSS are registered trademarks of Graphic Software Systems, Inc.



Graphic Software Systems

Contact this advertiser directly.

EGAs and VGAs, you're living dangerously.)

What of the split screen in text mode? It works fine. In fact, it not only resets the internal memory pointer to 0, but it also resets the text scan-line counter (which marks which line within the font you're on) to 0; so, the split screen starts out with a full row of text. There's only one trick with text mode: When split-screen pel-panning suppression is on, the pel-panning setting is forced to 0 for the rest of the frame. Unfortunately, 0 is *not* the "no-panning" setting for 9-dot-wide text—8 is. The result is that when you turn on split-screen pel-panning suppression, the text in the split screen won't pan with the normal screen, as intended, but will also display the undesirable characteristic of moving one pixel to the left. Whether this causes any noticeable on-screen effects depends on the text displayed by a particular application; for example, there should be no problem if the split screen has a border of blanks on both sides.

How Safe?

So, how safe *is* using the split screen? My opinion is that it's perfectly safe—although I'd welcome input from people with extensive split-screen experience—and the effects are striking enough that the split screen is well worth using in certain applications.

I'm a little more leery of horizontal smooth scrolling, with or without the split screen. Still, the Wilton book doesn't advise any particular caution, and I haven't heard any horror stories from the field lately—the clone manufacturers must finally have gotten it right. (I do remember some early clones that didn't get it quite right.) On balance, I'd say to use horizontal smooth scrolling if you need it. On the other hand, in fast animation you can often get away with byte scrolling, which is easier, faster, and safer. (I recently saw a game that scrolled as smoothly as you could ever want. It was only by stopping it with Ctrl-NumLock that I was able to be sure that it was, in fact, byte-panning.)

In short, use the fancy stuff—but only when you must.

In Conclusion

Well, I've used up all my space and more, as usual. Hal Hardenbergh has been teaching me about circle, ellipse, and tilted-ellipse drawing, and I'll present some fast circle-drawing code next time if I can get it done by then. (The trick is that Hal's code, while fast as all get-out, is written in 68000 assembly language. Keeping everything in the registers is considerably more of a challenge on the 8086, where the registers are not only too small but also too few, than on the well-endowed 68000.) As always, I'm open to your suggestions.

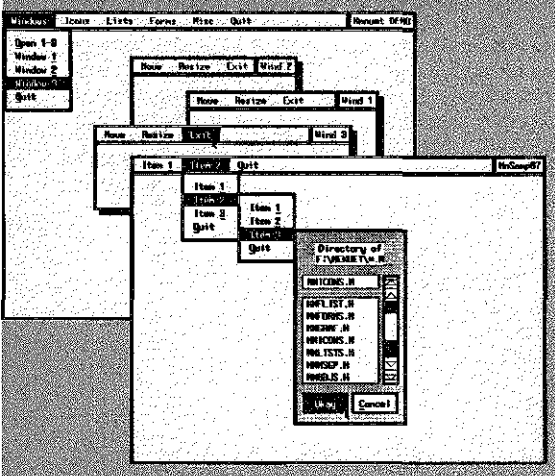
See you next time. PJ

Contributing Editor Michael Abrash is a senior software engineer for Orion Instruments, Inc., in Redwood City, California. Power Graphics Programming, *an anthology of Michael's "On Graphics" articles, published by Que Corporation and* PJ, *is available in bookstores and by mail order (see page 102). Michael is also the author of* Zen of Assembly Language, *published by Scott, Foresman & Company. Readers may write to Michael at 1599 Bittern Drive, Sunnyvale, CA 94087, or contact him via MCI Mail, user name* MABRASH.

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Last Word in Graphics Development Tools is from Ithaca Street Software!



....Menuet

Created to be object-oriented, extensible, and sophisticated looking, Menuet is one of the most thoughtfully designed GUIs (Graphical User Interfaces) now available for DOS applications.

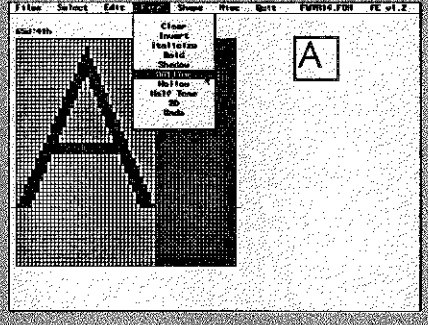
- Supports multiple windows, and windows can be moved and resized under mouse control.
- Supports virtually all GUI constructs, including pull-down menus, dialog boxes, scrollable lists, data entry forms, icon and button panels, more.
- Automatic screen save and restore for windows and forms using EMS memory, conventional memory, or disk.
- Text-style virtual windows, with automatic line wrap and scrolling, plus a traditional blinking cursor, all in graphics mode.
- Rubber-banded shape functions, including lines, rectangles, and ellipses.
- Functions to read graphics files in many popular formats, including PCX, IMG, and MSP.
- A complete development environment that provides over 350 callable functions.
- Includes the Interface Design Tool™ (IDT) for prototyping applications, with automatic source code generation.
- Only $250.00, No Royalties.
- Source code in C is available for an additional fee-- call for details.

....Icon-Tools

A unique product that provides the tools you need to create and display icons. Combines the best features of font and paint programs.

- Create icons with 2, 4, 8, or 16 colors in sizes from as small as you want to over 500 pixels wide by 400 pixels high.
- Up to 32000 icons per file, up to 10 files loaded by your application.
- Mouse-driven icon editor is better than most paint programs-- draw shapes, pattern fills, special effects, cut-and-paste, image manipulations, pixel masking, zoom, import graphics, resize icons, and much more.
- Includes dozens of sample icon files with over 500 high-quality icons, most in 16 colors (or convert them to monochrome if you like).
- Linkable function library for use with your application-- load icon files, then draw icons, animate icons, fade and scroll icons, create icon-based menus (with full mouse support), and more.
- Only $150.00, No Royalties (add $100 for source code).

Icon Ed. (above), Font Ed. (below)



....Font-Tools

We think icons and fonts are indeed different, so we also offer Font-Tools for creating and using fonts in graphics modes.

- Font editor with many editing functions, including facing for bold, italic, outlined, 3-D, and drop shadowed characters.
- Over 60 sample font files in a variety of type styles, with sizes ranging from 8 to 80 pixels in height. Use as-is or modify.
- Function library links with your application. Draw text with control of color, spacing, justification, direction, more.
- Utility programs for viewing, downloading, resizing, format converting, more.
- Only $150.00, No Royalties (add $100 for source code).

Ithaca Street Software, Inc.
1145 Ithaca Drive
Boulder, CO 80303, USA

Quality Tools for the Graphics Developer
Tel 303-494-8865 or FAX 303-494-7802
24 hr. BBS (12/2400 N,8,1) 303-494-8868

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from "VGA and EGA Split Screens," by Michael Abrash

1

LISTING 1

```
;
; Demonstrates the VGA/EGA split screen in action.
;
IS_VGA          equ     1       ;set to 0 to assemble for EGA
;
VGA_SEGMENT     equ     0a000h
SCREEN_WIDTH    equ     640
SCREEN_HEIGHT   equ     350
CRTC_INDEX      equ     3d4h    ;CRT Controller Index register
OVERFLOW        equ     7       ;index of Overflow reg in CRTC
MAXIMUM_SCAN_LINE equ   9       ;index of Maximum Scan Line register
                                ; in CRTC
START_ADDRESS_HIGH equ  0ch     ;index of Start Address High register
                                ; in CRTC
START_ADDRESS_LOW equ   0dh     ;index of Start Address Low register
                                ; in CRTC
LINE_COMPARE    equ     18h     ;index of Line Compare reg (bits 7-0
                                ; of split-screen start scan line)
                                ; in CRTC
INPUT_STATUS_0  equ     3dah    ;Input Status 0 register
WORD_OUTS_OK    equ     1       ;set to 0 to assemble for
                                ; computers that can't handle
                                ; word outs to indexed VGA registers
;***************************************************************************
; Macro to output a word value to a port.
;
OUT_WORD        macro
if WORD_OUTS_OK
        out     dx,ax
else
        out     dx,al
        inc     dx
        xchg    ah,al
        out     dx,al
        dec     dx
        xchg    ah,al
endif
        endm
;***************************************************************************
MyStack segment para stack 'STACK'
        db      512 dup (0)
MyStack ends
;***************************************************************************
Data    segment
SplitScreenLine dw      ?       ;line the split screen currently
                                ; starts after
StartAddress    dw      ?       ;display memory offset at which
                                ; scanning for video data starts
; Message displayed in split screen.
SplitScreenMsg  db      'split-screen text row #'
DigitInsert     dw      ?
                db      '...$'
Data    ends
;***************************************************************************
Code    segment
        assume  cs:Code, ds:Data
;***************************************************************************
Start   proc    near
        mov     ax,Data
        mov     ds,ax
;
; Select mode 10h, 640x350 16-color graphics mode.
        mov     ax,0010h        ;AH=0 is select mode function
                                ;AL=10h is mode to select,
                                ; 640x350 16-color graphics mode
        int     10h
;
; Put text into display memory starting at offset 0, with each row
; labelled as to number. This is the part of memory that will be
; displayed in the split-screen portion of the display.
```

2

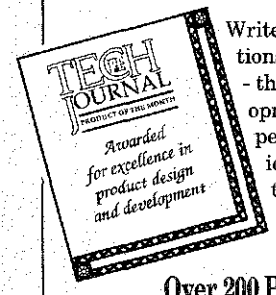
```
        mov     cx,25           ;# of lines of text we'll draw into
                                ; the split-screen part of memory
FillSplitScreenLoop:
        mov     ah,2            ;set cursor location function #
        sub     bh,bh           ;set cursor in page 0
        mov     dh,25
        sub     dh,cl           ;calculate row to draw in
        sub     dl,dl           ;start in column 0
        int     10h             ;set the cursor location
        mov     al,25
        sub     al,cl           ;calculate row to draw in again
        sub     ah,ah           ;make the value a word for division
        mov     dh,10
        div     dh              ;split the row # into two digits
        add     ax,'00'         ;convert the digits to ASCII
        mov     [DigitInsert],ax ;put the digits into the text
                                ; to be displayed
        mov     ah,9
        mov     dx,offset SplitScreenMsg
        int     21h             ;print the text
        loop    FillSplitScreenLoop
;
; Fill display memory starting at 8000h with a diagonally striped pattern.
        mov     ax,VGA_SEGMENT
        mov     es,ax
        mov     di,8000h
        mov     dx,SCREEN_HEIGHT ;fill all lines
        mov     ax,8888h        ;starting fill pattern
        cld
RowLoop:
        mov     cx,SCREEN_WIDTH/8/2 ;fill 1 scan line a word at a time
    rep stosw                   ;fill the scan line
        ror     ax,1            ;shift pattern word
        dec     dx
        jnz     RowLoop
;
; Set the start address to 8000h and display that part of memory.
        mov     [StartAddress],8000h
        call    SetStartAddress
;
; Slide the split screen half way up the screen and then back down
; a quarter of the screen.
        mov     [SplitScreenLine],SCREEN_HEIGHT-1
                                        ;set the initial line just off
                                        ; the bottom of the screen
        mov     cx,SCREEN_HEIGHT/2
        call    SplitScreenUp
        mov     cx,SCREEN_HEIGHT/4
        call    SplitScreenDown
;
; Now move up another half a screen and then back down a quarter.
        mov     cx,SCREEN_HEIGHT/2
        call    SplitScreenUp
        mov     cx,SCREEN_HEIGHT/4
        call    SplitScreenDown
;
; Finally move up to the top of the screen.
        mov     cx,SCREEN_HEIGHT/2-2
        call    SplitScreenUp
;
; Wait for a keypress (don't echo character).
        mov     ah,8    ;DOS console input without echo function
        int     21h
;
; Turn the split screen off.
        mov     [SplitScreenLine],0ffffh
        call    SetSplitScreenScanLine
;
; Wait for a keypress (don't echo character).
        mov     ah,8    ;DOS console input without echo function
        int     21h
;
; Display the memory at 0 (the same memory the split screen displays).
        mov     [StartAddress],0
        call    SetStartAddress
;
; Flip between the split screen and the normal screen every 10th
; frame until a key is pressed.
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;
FlipLoop:
        xor     [SplitScreenLine],0fffh
        call    SetSplitScreenScanLine
        mov     cx,10
CountVerticalSyncsLoop:
        call    WaitForVerticalSyncEnd
        loop    CountVerticalSyncsLoop
        mov     ah,0bh  ;DOS character available status
        int     21h
        and     al,al   ;character available?
        jz      FlipLoop ;no, toggle split screen on/off status
        mov     ah,1
        int     21h     ;clear the character
;
; Return to text mode and DOS.
        mov     ax,0003h        ;AH=0 is select mode function
                                ;AL=3 is mode to select, text mode
        int     10h             ;return to text mode
        mov     ah,4ch
        int     21h             ;return to DOS
Start   endp
;*****************************************************************
; Waits for the leading edge of the vertical sync pulse.
;
; Input: none
;
; Output: none
;
; Registers altered: AL, DX
;
WaitForVerticalSyncStart        proc    near
        mov     dx,INPUT_STATUS_0
WaitNotVerticalSync:
        in      al,dx
        test    al,08h
        jnz     WaitNotVerticalSync
WaitVerticalSync:
        in      al,dx
        test    al,08h
        jz      WaitVerticalSync
        ret
WaitForVerticalSyncStart        endp
;*****************************************************************
; Waits for the trailing edge of the vertical sync pulse.
;
; Input: none
;
; Output: none
;
; Registers altered: AL, DX
;
WaitForVerticalSyncEnd  proc    near
        mov     dx,INPUT_STATUS_0
WaitVerticalSync2:
        in      al,dx
        test    al,08h
        jz      WaitVerticalSync2
WaitNotVerticalSync2:
        in      al,dx
        test    al,08h
        jnz     WaitNotVerticalSync2
        ret
WaitForVerticalSyncEnd  endp
;*****************************************************************
; Sets the start address to the value specified by StartAddress.
; Wait for the trailing edge of vertical sync before setting so that
; one half of the address isn't loaded before the start of the frame
; and the other half after, resulting in flicker as one frame is
; displayed with mismatched halves. The new start address won't be
; loaded until the start of the next frame; that is, one full frame
; will be displayed before the new start address takes effect.
;
; Input: none
;
; Output: none
;
; Registers altered: AX, DX
```

Code continues . . .

Power Graphics for your PC!



Write powerful interactive graphic applications faster and easier using MetaWINDOW - the award winning leader in graphic development systems. MetaWINDOW, independently benchmarked the fastest graphics toolkit available, allows you to build tight custom, graphics applications without the massive bulk and "me too" look of other graphic systems.

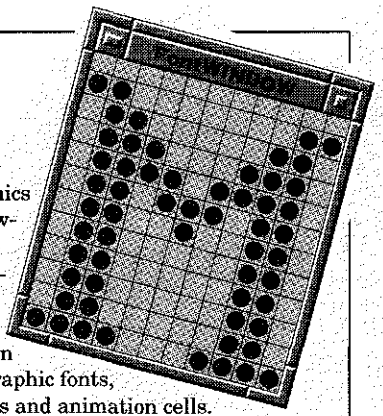
Over 200 Power-Packed Functions

- State of the art event driven operation.
- Ultra-fast bitmap and outline text drawing functions.
- Extensive rasterOp flexibility makes it easy to rubber-band or drag complex objects.
- QuickDraw superset speeds porting PC and Mac programs.
- Dynamic runtime support for 100+ popular graphics cards.
- Advanced versions available to support the full range of super-VGA and ultra-quick TI 34010 coprocessor cards.
- Full, illustrated documentation and free technical support.
- NO royalties!

MetaWINDOW (supports 15 popular C & Pascal compilers)	$250
(single language for Turbo C, Turbo Pascal or MS Quick C)	$125
FontWINDOW	$125

Free MetaWINDOW & FontWINDOW Working Demos

Download direct 408-438-5368 (1200/2400 baud, X-modem/Kermit), or call 408-438-1550 for free demo disk and complete capabilities brochure.



The most powerful PC graphics toolkit now has the most powerful font editing system! FontWINDOW makes it simple for you to create, edit & customize your own fonts. Using FontWINDOW you can create virtually unlimited graphic fonts, icons, foreign language styles and animation cells.

- More than 200 high quality fonts, icons, and foreign language styles - over 2 megabytes of fonts with NO royalties!
- Quickly merge characters from multiple fonts to create totally new custom designs.
- ZoomFont function makes it easy to scale fonts to virtually *any size. Includes ZoomFont C source code for scaling* fonts within your own programs.
- Full font format specifications, illustrated documentation and free technical support.

30 Day No Risk Money-Back Guarantee!

Order Hotline 800-332-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road, Scotts Valley, CA 95066
Phone: 408-438-1550 FAX: 408-438-5379

Number **143** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash code, continued

```
;
SetStartAddress proc    near
        call    WaitForVerticalSyncEnd
        mov     dx,CRTC_INDEX
        mov     al,START_ADDRESS_HIGH
        mov     ah,byte ptr [StartAddress+1]
        cli             ;make sure both registers get set at once
        OUT_WORD        al,START_ADDRESS_LOW
        mov     al,byte ptr [StartAddress]
        OUT_WORD
        sti_
        ret
SetStartAddress endp
;*************************************************************************
; Sets the scan line the split screen starts after to the scan line
; specified by SplitScreenLine.
;
; Input: none
;
; Output: none
;
; All registers preserved
;
SetSplitScreenScanLine  proc    near
        push    ax
        push    cx
        push    dx
;
; Wait for the leading edge of the vertical sync pulse. This ensures
; that we don't get mismatched portions of the split-screen setting
; while setting the two or three split-screen registers (register 18h
; set but register 7 not yet set when a match occurs, for example),
; which could produce brief flickering.
;
        call    WaitForVerticalSyncStart
;
; Set the split-screen scan line.
        mov     dx,CRTC_INDEX
        mov     ah,byte ptr [SplitScreenLine]
        mov     al,LINE_COMPARE
        cli             ;make sure all the registers get set at once
        OUT_WORD        ;set bits 7-0 of the split-screen scan line
        mov     ah,byte ptr [SplitScreenLine+1]
        and     ah,1
        mov     cl,4
        shl     ah,cl   ;move bit 8 of the split split-screen scan
                        ; line into position for the Overflow reg
        mov     al,OVERFLOW
if IS_VGA
;
; The Split Screen, Overflow, and Line Compare registers all contain
; part of the split-screen start scan line on the VGA. We'll take
; advantage of the readable registers of the VGA to leave other bits
; in the registers we access undisturbed.
;
        out     dx,al   ;set CRTC Index reg to point to Overflow
        inc     dx      ;point to CRTC Data reg
        in      al,dx   ;get the current Overflow reg setting
        and     al,not 10h ;turn off split-screen bit 8
        or      al,ah   ;insert the new split-screen bit 8
                        ; (works in any mode)
        out     dx,al   ;set the new split-screen bit 8
        dec     dx      ;point to CRTC Index reg
        mov     ah,byte ptr [SplitScreenLine+1]
        and     ah,2
        mov     cl,3
        ror     ah,cl   ;move bit 9 of the split split-screen scan
                        ; line into position for the Maximum Scan
                        ; Line register
        mov     al,MAXIMUM_SCAN_LINE
        out     dx,al   ;set CRTC Index reg to point to Maximum
                        ; Scan Line
        inc     dx      ;point to CRTC Data reg
        in      al,dx   ;get the current Maximum Scan Line setting
        and     al,not 40h ;turn off split-screen bit 9
        or      al,ah   ;insert the new split-screen bit 9
                        ; (works in any mode)
```


34010 Video Co-Processors from PC Tech

- Mono 2048 x 1536 displayed pixels
- Color 1024 x 768 displayed pixels
- Up to 5 M byte memory
- Optional CCITT imaging, TIGA, & DGIS software drivers
- C Compiler, Assembler, Linker, Debugger, DGIS Development Kit & TIGA Toolkit available
- PC/XT/AT compatible
- Custom designs available
- **PC Tech understands the 34010**

For further information, please contact:

PCtech
907 N. 6th St.
Lake City, MN 55041
(612) 345-4555 (Voice)
(612) 345-5514 (FAX)

PC Tech is a trademark of PC Tech, Inc.

PC, XT, AT, DGIS, and TIGA are trademarks or registered trademarks of their respective companies.

Number **208** on the Reader Service Card

Upgrade!
TurboGeometry Library

A Library of 2D & 3D Geometric Routines

New 3.0 upgrade! Twice as many 2D and 3D geometric routines (Over 300). Now with: Surfacing • Solids • Spherical Geometry • 3D Clipping • Tangents. Also includes additional HideLine algorithms • Intersections • Volumes • Areas • Transforms • Perspectives • Decomposition • and more. Easy to use manual. 30 day guarantee. $199.95 with source. Domestic S&H Incl. Foreign S&H Extra. TurboGeometry Library 3.0 is available in Turbo Pascal, Microsoft C, & Turbo C. Needs MS/PC DOS 2.0+. IBM/PC or Compatibles. We accept VISA, MasterCard, PO, Chk. USA negotiable funds only.

Disk Software, Inc.
2116 E. Arapaho Rd.
Suite 487
Richardson, TX 75081
USA
214-423-7288
214-423-4465 FAX
800-635-7760

Number **229** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
        out     dx,al   ;set the new split-screen bit 9

    else
    ;
    ; Only the Split Screen and Overflow registers contain part of the
    ; split-screen start scan line and need to be set on the EGA.
    ; EGA registers are not readable, so we have to set the non-split-
    ; screen bits of the Overflow register to a preset value, in this
    ; case the value for 350-scan-line modes.
    ;
            or      ah,0fh  ;insert the new split-screen bit 8
                            ; (only works in 350-scan-line EGA modes)
            OUT_WORD        ;set the new split-screen bit 8
    endif
            sti
            pop     dx
            pop     cx
            pop     ax
            ret
SetSplitScreenScanLine  endp
;*****************************************************************
; Moves the split screen up the specified number of scan lines.
;
; Input: CX = # of scan lines to move the split screen up by
;
; Output: none
;
; Registers altered: CX
;
SplitScreenUp   proc    near
SplitScreenUpLoop:
            dec     [SplitScreenLine]
            call    SetSplitScreenScanLine
            loop    SplitScreenUpLoop
            ret
SplitScreenUp   endp
;*****************************************************************
; Moves the split screen down the specified number of scan lines.
;
; Input: CX = # of scan lines to move the split screen down by
;
; Output: none
;
; Registers altered: CX
;
SplitScreenDown proc    near
SplitScreenDownLoop:
            inc     [SplitScreenLine]
            call    SetSplitScreenScanLine
            loop    SplitScreenDownLoop
            ret
SplitScreenDown endp
;*****************************************************************
Code    ends
        end     Start
```

LISTING 2

Because of lack of space, Listing 2 is not printed in this issue. It is available on the Code Disk for this issue as well as through our bulletin board services (see pages 54 and 96). —Ed.

PRINTED GRAPHICS IN HIGH RESOLUTION!

GRAPHLINK™

THE PRINTER GRAPHICS SOLUTION™

GraphLink is the Printer Graphics Toolkit that does for your printer what other graphics packages do for your screen -- at the printer's resolution! *GraphLink is the solution you've been waiting for because GraphLink . . .*

- Does *NOT* do a low resolution screen dump -- it builds its own image at the *printer's* resolution - *even at 300 dots-per-inch for a LaserJet!*
- Provides more than 70 routines which closely emulate Borland's BGI graphics commands, including line, bar, ellipse, arc, pie slice, area fill, text, and viewport commands.
- Works with the most popular printers - including the HP LaserJet and DeskJet, and Epson, IBM, Toshiba, NEC, and other 9 and 24 pin dot matrix printers.
- Supports Turbo C, Microsoft C, Turbo Pascal, Quick Pascal, TopSpeed Modula-2, and other popular compilers.

Includes an excellent manual, free technical support, and a 30 day money back guarantee, all for just $125 (or $225 with source code)! (Foreign/Canada orders please add $10/$5.) *Order today, and solve your printer graphics problems!*

Visitech Software (412) 733-4775

D5 3807 Ridgewood Ct., Pittsburgh, PA 15239 USA

Number **304** on the Reader Service Card

CGL3D New software

A Complete 3D Color Graphic Library for your PC

Create powerful 2D/3D graphic application programs, display 2D/3D spreadsheet and experimental data with functions of an easy to use graphic library.

*2D/3D Point, Line, Text, Polygon, Fill, Surface
*World coordinates, multiple viewports
*True 3D perspective
*Text editor
*Element color/type
*Zoom, Transformation
*2D/3D Views
*2D/3D Rotation
*2D/3D Conversion
*PC MOUSE interface
*Plot 3D surface data
*IBM ROM BIOS interface

*Example source code with easy to learn user manual
*The Library functions can be access by FORTRAN or C programs. Library is written in FORTRAN 77 and compiled with Microsoft FORTRAN compiler V4.

Similar in most functions to Tektronix PLOT-10(TCS) mainframe graphic plus additional true 3D functions

To order, Please send $175 plus $3 for shipping to:
F.G.SOFT
P.O. BOX 2152,
Hawthorne,CA 90251-2152

Get user manual and demo flopy disk for $10, refundable upon purchase

Tel 213-542-5461

Number **307** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Simple Terminal Program for Windows

by William S. Hall

Little information has been published about the communications interface of Microsoft Windows. In this article, I will show how to use its most important communications functions by presenting a simple terminal program. As the display is handled by the TTYCLS functions discussed in earlier *PJ* articles (see references), only a teletypewriter is emulated. Nevertheless, the program contains several features of interest, such as window subclassing; opening, setting, and closing a communications port; polling techniques; and font enumeration. And, because it was necessary to change the tty display functions slightly, this month's code disk also includes those previously distributed Windows programs that were affected.

Windows Communications Driver Bug

If you are using Windows 2.10 or 2.11, you need to be aware of a bug affecting Xon/Xoff flow control in the communications driver COMM.DRV. Windows 2.03 is not affected, nor are early issues of Windows 2.10.

Among a number of symptoms, the most obvious is receiver-buffer overflow. To see it, run Windows Terminal, select Xon/Xoff handshaking, log in to a remote machine, begin a long typeout, and then activate any menu and leave it down. The display will stop, of course, but the driver's receive queue will continue to fill. If the bug exists, an Xoff will never be sent to the host. When you release the menu (after waiting long enough for the queue to fill), you will see that you have missed some text.

After studying the source for COMM.DRV, I found the problem in the interrupt routine: A flag indicating that an Xoff must be sent was not saved under certain conditions. The oversight was reported to Microsoft, and an updated driver is now available on various bulletin boards, including ONLINE. I have included a copy with WNTERM.

Running WNTERM

To use the program, copy WNTERM.EXE to a suitable directory and run it under Windows. WNTERM first tries to open COM1. If the attempt is unsuccessful, it tries COM2; if this fails, the program exits. If a port can be opened, the program sets the baud rate, parity, and word size according to the COMx: strings in the [ports] section of win.ini.

You may want to change these values if doing so does not interfere with the use of a serial printer or some other more important usage. You can use the Control Panel's Setup menu, or you can edit win.ini directly. Be sure, however, that the strings exist in win.ini—the program may not load at all if they are missing. If in doubt, look for an entry such as

```
[ports]
; other information ...
COM1:=1200,n,8,1
COM2:=9600,n,8,1
```

If you prefer not to make the settings permanent, you can always use WNTERM's Communications menu to set the desired parameters.

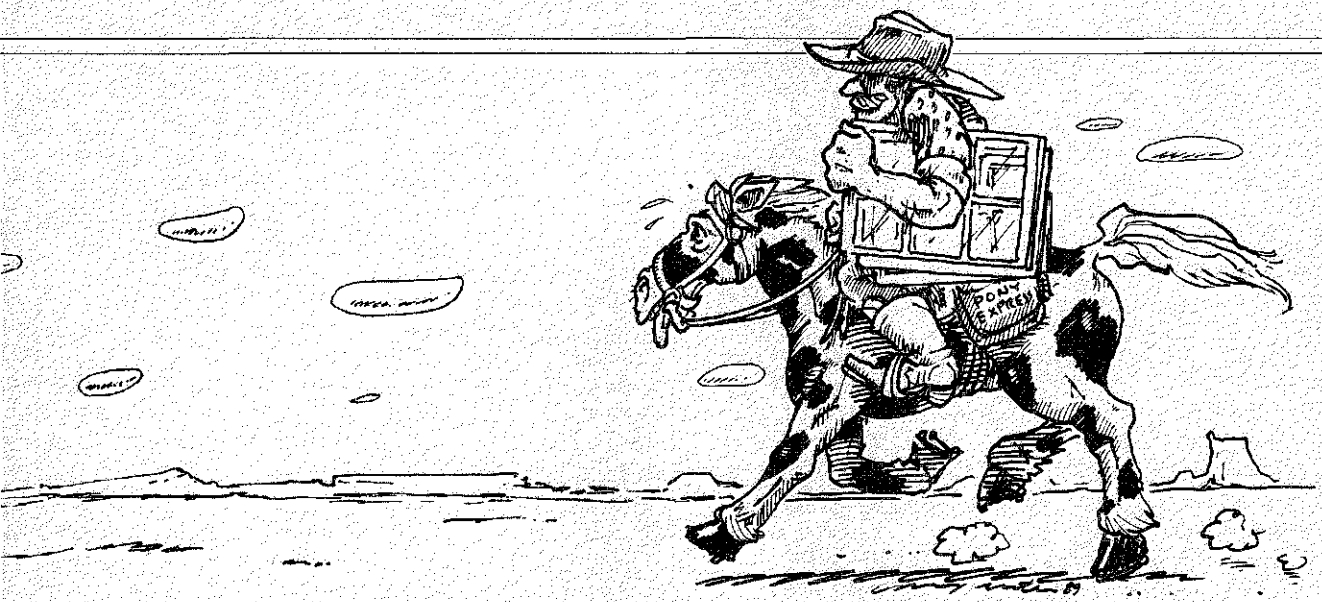
When the window appears, the program will be ready to communicate with the port shown in the title bar. You can switch between line and local operations by toggling the Line/Local menu item. At any time, you can clear the screen with the ClearScreen menu choice. The Scroll Lock key will start and stop the display.

After the program has run for the first time, an entry in the following form is made in win.ini:

```
[Wnterm]
SmallFont=0
```

If you change 0 to 1, the program will examine all Courier fonts installed on your system. If a font is found

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



with a height less than that of the Windows system font and a width less than or equal to that of the Windows system font, then that font will be selected. In this way, it is possible to have more lines on the screen. If you install COURA.FON, you can have as many as 73 lines on a VGA screen and 53 on the EGA. (COURA.FON is normally copied to your Windows directory only if you have a CGA; you may have to retrieve it from the distribution disks.) If no small font is found, then the system font is used. You will then have a maximum of 29 lines on the VGA and 32 on the EGA. (The EGA system font is only 10 pixels high, whereas on the VGA it is 15.)

If you are using Windows 2.10 or 2.11, you need to be aware of a bug affecting Xon/Xoff flow control in the communications driver COMM.DRV.

Comments on the Small-Font Feature

A few weeks ago I visited Video-7 in Cupertino, California, where one of the engineers lamented to me on the lack of terminal emulators in MS-DOS that provided a lot of visible information on the screen. Of course, he was aware that many programs feature memory paging, but what he needed was to see as many lines as possible when using an auxiliary terminal for debugging. So, I added the small-font capability to this program, and I have heard since that WNTERM has been of some value to him. As a consequence, you will see an example of how fonts can be enumerated and chosen based on some criterion.

Message Loop

In earlier articles, I mentioned the problem of not having a separate thread of execution in Windows for continuous processes. In terminal emulation, it is necessary to poll the communications port for the incoming character stream and move it to the screen. But in basic windowing, the program itself has nothing to drive this action. Most of the time, the program is waiting for messages; if no messages are sent, nothing happens. Although the communications driver is interrupt driven, it posts no message that characters are present to be read. Hence, the basic message loop of Windows must be modified to poll the communications port if there are no other messages and to retrieve and show any pending input.

Listing 1 shows WNTERM's **WinMain** and **ProcessComm**. After initialization, the program begins a **PeekMessage** loop. If there are no messages for the window (and if it is not iconic), **ProcessComm** is called to read the communications port and dispatch any characters found to be displayed.

In a standard Windows message loop, **GetMessage** returns 0 when a WM_QUIT message has been received and the program drops out of the loop. **PeekMessage**, however, returns 0 whenever it has no messages, and so a separate check is needed to see if WM_QUIT has been received. Also, **PeekMessage** can be instructed to leave messages in the queue or process a limited range. Here, we simply pull and dispatch all messages.

ProcessComm is called if the main window is not iconic. Here, you see one method of reading the communications port without its becoming blocked. First, the status of the port is checked with **GetCommError**. In particular, we are interested in determining the number of characters in the communications driver's receive queue. Commensurate with space in the program's local buffer, these are read and sent to the window with a **PostMessage** call via a WM_USER message.

An alternative approach to reading the port is to simply read into the local buffer as many characters as can fit. Then, if an error occurs (signaled by a negative return value), a call to **GetCommError** is made, since now the port is blocked and can be freed only by this

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

call. However, this method occasionally still leaves the port blocked. Since the absolute value returned by **ReadComm** contains the number of characters read, if errors are not particularly important (as they certainly are not here), then the code of Listing 1 is adequate.

Some provision has been made for the case where the local buffer still contains characters read from the port but not yet displayed. A global variable, *Buflen*, is used to index into the local buffer. Of course, since the buffer itself is linear and not circular, it is necessary to move any unread characters back to the beginning before loading it again by **ReadComm**. If you look at the **SubClassWndProc** function, case WM_USER, in Listing 2, you will see where the move is effected.

Although it may seem crude to use a linear buffer and move unused characters around, the situation occurs infrequently, and the overhead of a circular queue is avoided. After the local buffer is updated and made ready for processing, a WM_USER message is posted to the window. This message thus appears in the queue, and so the next time **PeekMessage** is called, **ProcessComm** will not be executed. Instead, the buffer is emptied to the display. In this way, synchronization is maintained between writing from and reading to a resource (buffer) "shared" between two "threads" of execution (processing a message or processing the communications port). You may want to compare the action here with that of PMTERM, the Presentation Manager version of WNTERM (see *PJ* 7.4).

Line Switch

Real terminals have off/on line switches. When the unit is offline, transmission and reception through the communications port is suspended and keyboard input is sent to the display. When the unit is online, keyboard characters are sent to the port and input from the port is displayed. However, if local echo is on, the keyboard input also shows up on the screen. Clearly, a close relationship exists between local echo and line state.

This effect can be simulated quite nicely in Windows using *subclassing*, which is a means of modifying the behavior of a window class by altering its window procedure. In WNTERM, when the unit is offline, **MainWndProc** is the window procedure. When the unit is online, all messages from Windows first go through **SubClassWndProc**, which itself is chained to **MainWndProc**. The action of making and breaking the chain occurs in response to the online/offline menu selection.

Listing 2 shows the main features of these two procedures. With all the noise removed, here is what is happening:

```
SubClassWndProc(...)   // come here first if online

    switch(message) {

        case WM_USER: // show characters obtained
                      // from comm port
            TTYDisplay(Buffer);
            break;

        case WM_CHAR: // write keyboard characters
            WriteToCommPort(char);
            if (!LocalEcho)
                break;
```


SOFTWARE DEVELOPERS: ARE YOUR PRODUCTS TOO HARD TO INSTALL?

In today's competitive market, your product **must** be easy to use -- immediately. It is no longer commercially acceptable to tell the end-user to use batch files, COPY, DISKCOPY, RESTORE, or to require wading through paragraphs of text in the manual. The installation process is something that a program developer **should** and now **can easily** completely automate.

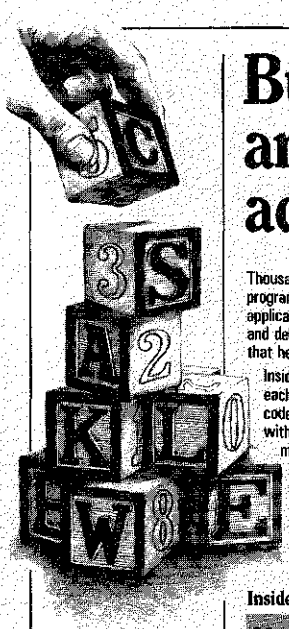
INSTALL 2.3 is a bullet-proof automatic installation program you can distribute - royalty free - with your product • Features: Up to 4.3 gigabyte file sizes • Windowing/color user interface • **Full C source** with project, make, link, and batch files • File integrity verification • Intelligent handling of virtually all errors (open disk drive doors, full disks, unformatted disks, etc) • Hard & floppy disk installation • Multiple source & target disks • Multiple target subdirectories (which *INSTALL* will create if necessary) • Unlimited free tech support • Granular installation • Easy method to add-in new functionality • Ability to execute DOS commands • Can intelligently modify CONFIG.SYS & AUTOEXEC.BAT files • Includes **data compressing file librarian** • **NEW** Automatically detects and can base the installation on the CPU (86/286/386), DOS version, display (CGA/B&W/MONO/EGA/VGA), NetBIOS, ANSI.SYS, LIM, etc.

INSTALL offers 3 years of proven reliability that has been used to install over 500,000 copies of products including some of the most prestigious in the industry. We would like to add your name to that list.

ORDER NOW TOLL FREE 1-800-331-2783

KNOWLEDGE DYNAMICS CORPORATION
Highway Contract 4 Box 185-H, Canyon Lake, TX 78133-3508
Voice: 1-512-964-3994 FAX: 1-512-964-3958
MasterCard/VISA/COD/Purchase Order **$149.95**

Number 295 on the Reader Service Card



Build with an Inside! advantage...

Thousands of C, BASIC, FORTRAN and Modula-2 programmers are already using Inside! to improve application performance. You have the best compiler and debugger, now add the performance analysis tool that helps create world class applications!

Inside! analyzes your program in real-time, timing each instance of a function, source line or arbitrary code fragment, all with microsecond accuracy and without source code modification. Utilize the following modes to know where to focus your optimization efforts for maximum reward.

- Function/procedure timing
- Call target timing
- Source Line timing
- Arbitrary event timing
- Non-executed functions or source lines
- MS-DOS function timing
- Overlay and assembly language support

Inside! is available for popular compilers . . .

Call or write today for a free brochure and the latest list of Inside! products.

To Place Orders
(800)537-5043
Product Support
(508)478-0499
Visa/Mastercard/C.O.D. Accepted

$125.00 EACH
Satisfaction guaranteed

Paradigm Systems
P.O. Box 152 Milford, Massachusetts 01757
Inside! is a trademark of Paradigm Systems.

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        default:        // chain to main window
                        // procedure
                return (CallWindowProc((FARPROC)
                        fpMainWndProc));
    }
    return (0L);
}

MainWndProc(...)        // come here first if offline

    switch(message) {

        ....            // other messages

        case WM_CHAR:
            WriteToScreen(char);
            break;

        return (DefWndProc);
    }
    return (0L);
}
```

Hence, when the unit is offline, a keypress simply causes the appearance of a WM_CHAR message, which is fielded by **MainWndProc**, and results in a character appearing on the application's window. But when the unit is online, WM_CHAR is first handled by **SubClassWndProc**, which immediately sends the character to the communications port. Then the local echo flag is tested. If the flag is set, the WM_CHAR message is passed on to **MainWndProc**, which writes the same character to the screen.

Switching the window procedure is accomplished in the Line/Local menu. You can see the details in the excerpt of **WndCommand** at the top of Listing 3. **SetWindowLong** is called to set the appropriate window procedure, and the menu is changed and redrawn to give the effect of a toggle. The **CallWindowProc** function used in **SubClassWndProc** requires a procedure instance of **MainWndProc**, however; this is created during initialization.

Choosing a Small Font

Windows provides a built-in font, called the System font, using the ANSI character set. A modern typeface having a fixed pitch, its size varies with the display. In addition, a number of typographic fonts of various sizes are available; these include Courier, Helvetica, and Times Roman. Which font is installed during setup depends on the characteristics of the display and the printers selected. Of course, other fonts can be added later using the Control Panel.

If a font is installed, it can be examined by a program for its characteristics, which are many—more than 20 fields are available in the TEXTMETRIC structure alone. TEXTMETRIC is employed to describe parameters of a physical font; LOGFONT, on the other hand, is used to define the attributes of an object that draws text on the display. A LOGFONT structure is passed to **CreateFontIndirect**, which returns a handle to the physical font most closely approximating (according to a rather complicated criteria) the logical attributes specified. This handle can then be used in showing text.

Sometimes it is practical in an application to simply specify a set of desired font attributes and accept whatever is returned. In other cases, it is useful to enumerate the physical fonts available before creating

C++ *Vector and Matrix Classes*

++ Excellent introduction to Object Oriented Programming techniques
++ Full Source Code included
++ Complete Manual/Users Guide
++ Numerous examples and test programs

- Overloaded arithmetic operators and math functions for Double, Float, Int and Complex Vectors and Matrices
- "Slice operators" for high efficiency
- Real and Complex FFT and Inverse FFT "Server" Classes
- Compiles under MS-DOS (Zortech) or Unix (GNU "g++" or AT&T "cfront")

Only $99

($3 shipping, WA residents add 8% sales tax)

Rogue Wave Associates
Post Office Box 85341
Seattle WA 98145-1341

Tel: (206) 523-5831
"The experts in Object-oriented numerics"

Number 324 on the Reader Service Card

Introducing C talk / Views™

Reduce C Programming for MS Windows by 75%

Introducing C_talk/Views™, a major productivity tool for Microsoft® Windows program development in C. Slash your MS Windows program development time by up to 75% through the powerfull object-oriented programming (OOP) technology of C_talk. With C_talk/Views, the most common elements of all MS Windows programs are packaged into reusable software components-object classes. Over forty software components are included as a complete applications development framework. These object classes provide an easy-to-learn and powerfully consistent interface to the rich functionality of MS Windows. C_talk/Views is also a real OOP environment. Use the software components as they are, or extend them to meet the most demanding applications.

Coming soon...C_talk/Views for the Apple Macintosh®... now your MS Windows program will port to the Mac with little or no change!

$450.00 In the US, add $5 for shipping. ($25 for Int'l handling)

CNS, Inc.
Software Products
7090 Shady Oak Rd.
Minneapolis, MN 55344
612-944-0170, Fax 612-944-0923

. . . providing and advancing object-oriented methodology.

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the corresponding logical font. To this end, Windows uses its ubiquitous callback technique. **EnumFonts** is called with a pointer to a face name (Courier, for example), a far pointer (created with **MakeProcInstance**) to the callback function (which must be exported), and any user-supplied data. This function is then called repeatedly by Windows with both TEXTMETRIC and LOGFONT structures until there are no more fonts to enumerate or the callback function returns 0.

WNTERM requires a fixed-pitch font. Of the available standard Windows typographic fonts, only Courier is fixed pitch; so, this is the face name passed to **EnumFonts**. In the callback function, a simple test is made to see if the enumerated font is smaller in height than the system font (the width and height of the system font are the user-supplied data); if it is, the logical font structure is copied and saved. The process is continued until all fonts have been examined. Then, if a smaller font has been found, the LOGFONT structure is used to get the handle. The details are shown in Listing 3 in **WndCreate**, where **EnumFonts** is called, and in **FindSmallFont**, the callback function.

Using the Communications Interface

It is a pleasure to use the Windows serial communications interface. Fifteen functions are available, covering a wide range of options. Basic usage during the life of a program is as follows:

1. Open either COM1 or COM2 and set the queue sizes.
2. Obtain, modify, and set the port's device control block (DCB).
3. Read and write to the port.
4. Close the port.

Listing 4 shows a part of the initialization code where the port is opened and set. After the port has been opened successfully with **OpenComm**, the DCB is obtained with **GetCommState**. Then, the "ports" section of win.ini is read to obtain a string containing the baud rate, parity, word size, and number of stop bits. This string is then concatenated to the port name to obtain a *DOS mode string*. The results are translated by **BuildCommDCB** to set the DCB's basic parameters. Then, all the fields needed to enable both outbound and inbound Xon/Xoff flow control are initialized. Finally, **SetCommState** is called, and the port is ready for use.

The use of **BuildCommDCB** is simply a convenience for setting parity, baud rate, word size, and stop bits when presented with a mode string. Of course, theses values can be set directly from numerical data without using this function call.

We have already discussed one way of reading the port. In this program, writing to the port is done one character at a time, and so no particular cautions are necessary to prevent overflow of the send buffer. However, when writing a lot of data (as in file transfers, for example), it is necessary to check for possible overflow by first using **GetCommError** to read the number of characters currently awaiting transmission and possibly to wait until enough space is available in the write

NEW BBS

SCREEN MANAGER
P·R·O·F·E·S·S·I·O·N·A·L

ADVANCED WINDOWING

• The Screen Manager Professional gives your applications the finest bullet proof windowing interface available. Although other products may boast of hundreds of functions, they often involve needless repetition. Our functions are elegant and powerful. Each accomplishes a variety of tasks with one command. All window functions are written in hand-optimized assembly language to provide the fastest possible response and smallest possible size.

ORDER

•30 day Money Back Guarantee
•Screen Manager Professional $250
•Source Code $100
•Specify Compiler when ordering •Include UPS shipping: ground $5, overnight $20, 2nd day air $10, outside U.S. $30, COD $5. •Visa, Mastercard, check or money order. •Payment must be in U.S. dollars drawn on a U.S. bank. •PA residents add 6% sales tax.

POWERFUL

• Amazing Speed
• Incredible Menus
• Context Sensitive Help Editor
• OS and Compiler Independent
• Window Functions are 100% Assembly
• Keyboard Filtering for Data Entry Systems
• Full Video Adapter Support
• Compact Linkable Library
• Background Processing
• Reconfigurable Memory Allocation Function

VERSIONS AVAILABLE FOR

•Microsoft C •Zortech C++
•Quick C •Turbo C •Lattice C •
Watcom C

AMAZING

"...an amazingly complete collection of window commands." "The support functions are easy to use and very powerful. This library warrants serious consideration."
Computer Language-May '89

• Free yourself to concentrate on the substance of your program. Over 130 functions provide all the windowing tools needed to develop serious applications whether you are a professional or a hobbyist.
• The Screen Manager Professional can help you create easy to use, visually stimulating, programs with minimum effort. Now powerful windowing is available to you at an affordable price.

NEW BBS

Demos, Software, Tech Support- A forum for the Professional Programmer
BBS (814) 234-1881

LAI **Logical Alternatives Incorporated**
Calder Square, P.O. Box 10674
State College, PA 16805-0674
(814) 234-8088 FAX (814) 234-6864

Number **246** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

queue. (A wait should be polite to other applications. I have found that the **Yield** function can be helpful in allowing other Windows programs to run while your program waits for completion of some event.)

Although it's not needed in WNTERM, it is possible to set an event mask in the communications driver that can be used to test for a number of interesting occurrences. Some of these are break detection, state changes of modem control lines, event character detection, notification of an empty transmit queue, and new character in the receive queue. Nevertheless, there is a major oversight: Although you can get an indication that the state of a modem control line (Data Terminal Ready, Clear To Send, Ring Indicator, Carrier Detect) has changed, you cannot read the state itself. This lack of information is surprising, since both the state as well as the state changes can be obtained together in the hardware. I experimented by modifying the communications driver to return the actual state of the modem lines in the top four bits (currently unused) of the event word. This feature has proved especially useful in testing for loss of carrier. Microsoft should consider adding this option to the Windows communications driver.

Additional operations include enabling and clearing break state, flushing send and receive queues, and controlling output modem control lines, such as DTR and RTS. Finally, the DCB itself has more than 30 fields that can be used to determine the behavior of the communications interface. Not all, however, seem to work. For example, a field called TxDelay is supposed to determine the minimum amount of time that must elapse before sending the next character. However, it has no effect, although its proper operation would be useful in accessing some public networks where certain character sequences require a specific intercharacter delay. Currently, you have no choice but to use the old-fashioned way of inserting a delay (with, of course, an appropriate use of a **Yield** call).

Many of the fields in the DCB structure are bit fields that are defined on bytes rather than on the currently accepted integer types. Hence, if you set warning level 3 when compiling a Windows program, you will get an avalanche of warning messages unless you define NOCOMM before including windows.h. Of course, if you need the communications functions and you like to use as much error and type checking as possible, you have no choice.

Miscellaneous Considerations

WNTERM has a simple dialog box that allows the user to select important communications parameters. The principal controls used therein are radio buttons clustered together according to function within group boxes. In the PM version of this program (*PJ* 7.4), you saw how easily automatic radio buttons are handled. Windows lacks this feature, and more local processing is needed to achieve the same results.

You control scrolling of text in WNTERM from the keyboard by toggling the Scroll Lock key. Each time it is pressed, the key's state is read by an interception of the WM_KEYDOWN message and a setting of a flag (see Listing 2). If the flag is TRUE, then the communications port is not read and display stops. The communications

Software Tools

For Programmers & Non-Programmers

Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support. User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149.

On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing

P.O. Box 678 / Zephyr Cove, NV 89448

(702) 588-3737

Number 16 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROSOFT LANGUAGE PROGRAMMERS!
MIXED LANGUAGE WINDOWS ARE HERE!

MXWINDOWS MULTILANGUAGE WINDOWING LIBRARY SYSTEM

Fast, Powerful, Versatile Windowing Actions You Can Call From BASIC, C, Pascal, FORTRAN, QuickBASIC, QuickC, and QuickAssembler

- Exclusive "TAILORING" concept provides FORMATTING AND COLOR CONTROL OF EACH TEXT LINE AND ELEMENT. Create MULTIPLE WINDOW Setups.
- Select from a VARIETY OF WINDOW DESIGNS - Standard, PopUp, ZoomUp, SlidePanel, FanOut, and others.
- SAVE/RESTORE UNLIMITED FULL or selctd screen areas - RESTORE SELCTD SCREEN AREAS ANYWHERE!
- USER PROMPT DEVICES - MENUBAR, MENUARROW, FIRST LETTER HIGHLIGHT.
- In System handling of USER MENU SELECTIONS - Cursor Line, Key Press info. returned.
- Create FULL BLOWN TOP SCREEN MENU SYSTEMS with P/Downs, etc.
- BAR MENUS on ANY row from 1-25.
- JUSTIFY Window Text (Option).
- SPECIAL EFFECT Screen/Window Clearing; explode, UpDown, etc.
- Variety of BORDER, HEADER, and 3-D SHADOW EFFECTS.
- Create and INTERMIX invoice, entry, and other forms with menus, messages, etc.
- INCLUDE POWERFUL WINDOWED WORK STATIONS IN YOUR PROGRAMS LIKE WINDOWED DIR. READ/SCROLL, USER EQUIP. QUERY, AND OTHERS!
- READ Video, Hardware, Disk, Drive, Equipment Stats with powerful BIOS/DOS Interface.
- Comprehensive CALENDARING SYS. Procedures to inc. in programs.

AND MUCH MORE...!

SPECIAL INTRODUCTORY OFFER

Order MXWINDOW BEFORE 12/31/89 and RECEIVE FREE, MXTASKS, A POWERFUL LIBRARY that lets you INCLUDE Operating Work Station Systems in your programs:

- Windowed Mini-Editor.
- ScrollText - Read/Scroll or send selected window text to disk or printer.
- DOS Tasks Work Station.
- Filter User Inputs.

AND MORE...

MXWINDOW & MXTASKS Libraries:

$59.95

Comprehensive Manual / Reference & Source Code Sample Modules on disk to reduce cost.

(516) 434-1979

Send check or money order to:
MTD Software Development, LTD
21 Ralph Avenue
North Brentwood, NY 11717

Microsoft, "Quick" as used, are Registered Trademarks of the Microsoft Corporation

*See your MS manual for Mixed Lang. Compatibility

Number 312 on the Reader Service Card

buffer fills to its limit. Along the way, Xoff is sent. Hence, as long as there is some flow control between WNTERM and the host, no characters are lost.

If you want to make use of the special keys such as Scroll Lock, function keys, and arrow keys, you must either field them with WM_KEYDOWN or use an accelerator table and process them as WM_COMMAND messages. Most special keys have defined virtual key codes. Strangely, the value for Scroll Lock (0x91) is not listed in windows.h.

What is usually called the "cursor" on a terminal is known as the "caret" in Windows. The caret is a shared resource, and so a Windows program can create it only when a WM_SETFOCUS message is received; similarly, when WM_KILLFOCUS is seen, the caret must be destroyed. In addition, whenever a WM_SIZE message is received, it is necessary to recreate the caret *if* the window has the focus. Listing 2 shows the details.

Before text can be displayed, the caret must be hidden. After the text has been written to the screen, the position is updated and the caret is again shown. If the caret is not hidden before drawing text, you will see the track of the caret across the screen.

When a WM_CHAR message is received, the low word of *lParam* contains a repeat count of the character. In this simple program, we just call the handler for one character that number of times. A more sophisticated approach would be to load a local buffer before calling the display code. And, since having a count greater than one seems to be fairly rare, it would be worth the effort to have separate code for the case of a single character and for counts greater than one.

Naturally, you must close the COM port before exiting the program, since it is also a shared resource. It is less obvious that you should also close the port when ending a Windows session. If you do not, reexecuting Windows without rebooting the system will result in an error when an attempt is made to open the port. A way to handle all situations can be seen in the WM_CLOSE and WM_ENDSESSION messages in Listing 2.

Conclusion

I have been asked many times why I prefer to use Windows rather than MS-DOS to write primarily character-based applications. The answer is really quite simple: Windows presents an operating system that controls all the machine's resources and that has a rich applications programming interface (API). Although it is certainly interesting to develop device drivers, screen managers, and terminate-and-stay-resident programs (TSRs) to fill the gaps in MS-DOS, I prefer to be able to write an application that provides a standard presentation to the user while not having the additional burden of creating an adequate software interface to the hardware.

Windows also provides a set of standards for data exchange among applications as well as a common user interface. Finally, Windows is multitasking, although its success here depends on each running program yielding the processor in a timely fashion. In short, programming in Windows is interesting, and often quite challenging.

Nevertheless, I continue to be surprised at the dearth of Windows programmers. There seems to be a certain snob appeal in *not* getting involved in a WIMP (WIndows, Menus, Pointing devices) system. And it *is* painful to get started. But Windows is precisely the sort of area that should strongly interest the very best in this business. Now that you can "buy in" reasonably cheaply (the Software Development Kit is frequently discounted by nearly 50 percent), more of you may try it and perhaps even like it. PJ

William S. Hall is a senior engineer at CONNECT Inc. in Cupertino, California. He specializes in Windows communications software.

Code follows.

References

"Smart Terminal Emulation in Windows." *Programmer's Journal* 7.1 (1989), 74–83.

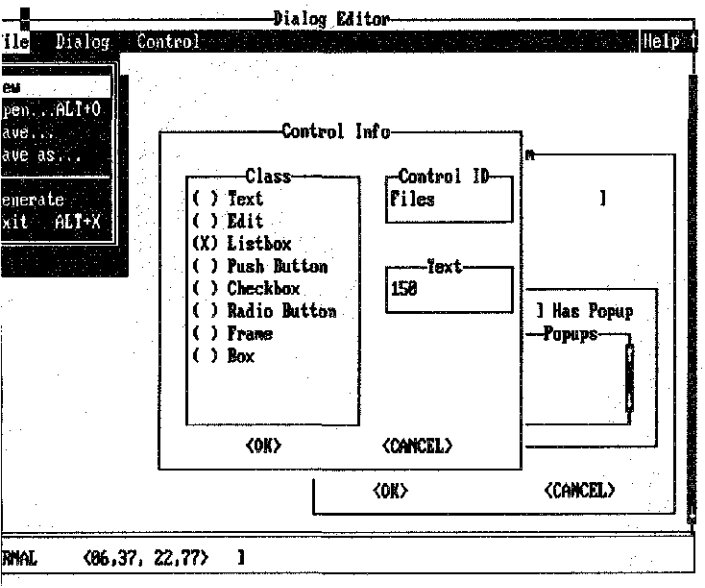
"A Simple Debugging Utility for Presentation Manager and Windows." *Programmer's Journal* 7.2 (1989), 56–73.

"A Substitute for the Missing TYPE Command in Windows." *Programmer's Journal* 7.3 (1989), 64–73.

"A Simple Terminal Program in Presentation Manager." *Programmer's Journal* 7.4 (1989), 32–43.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Finally, the windowing library which everyone has been waiting for! Enter the 1990's with.....



MEWEL

(Magma Extensible Window and Event Library)

- **Works on message passing, like Windows, Presentation Manager, the Mac, and X Windows.**
- **Has many of the same messages and functions as Windows and PM. Easy to port your apps to and from MEWEL.**
- **Object Oriented. Create your own window classes, or modify existing ones by "subclassing". Send messages to a window!**
- **Full source code available!!! Customize MEWEL even further! Use your standard debugger!!!**
- **Supports Windows-like controls, such as dialog boxes, scroll bars, list boxes, push buttons, radio buttons, and check boxes.**
- **Full mouse support! Click away on menus, windows, or controls.**
- **Pull-down menus which you can modify dynamically. Check, disable, and grey an entry.**
- **Dialog editor included with full source code.**
- **Supports multiple timers and multiple monitors.**
- **Create SAA compatible applications.**
- **Comes with libraries for Microsoft and Turbo C. (large and medium models)**
- **No royalties!!!**

y our functional demo disk. Call the Magma Systems BBS ow, or send us a check for $15. You have nothing to lose and erything to gain!

NEW in version 1.2

MS-Windows similar resource compiler.
Multi-line edit fields.
Formatted fields
User-defined classes.
OS/2 version available

INTRODUCTORY OFFER

MEWEL with libraries, dialog editor, and sample apps	$150!
MEWEL with the above AND FULL SOURCE CODE	$295!

Shipping and handling. $5 in US, $20 overseas. (Sorry, no credit cards.)

News From MAGMA SYSTEMS

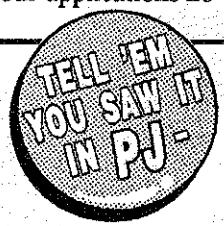
ME TEXT EDITOR : We have added a full **UNDO** and **REDO** capability to ME, and the ability to swap out the entire editor to disk for compiles. In addition to our DOS and OS/2 versions, we have released a **XENIX** version, so you don't have to suffer along with VI anymore! This in addition to ME's standard features: multiple tiled windows, multiple buffers, regular expressions, EGA, VGA, and other monitor support, CTAGS, multiple keystroke macros, column operations, capture of a DOS session, discontiguous line operations, and our famous C-like macro language! Of course, full source code is available so you can add full text editing capabilities to your applications. DOS and OS/2 versions: only $89, $189 with source. The XENIX/UNIX version is $150 without source, $295 with source.

Netlogic 3270 Emulator for MS-Windows and PM - powerful, fast emulator allows you to access your mainframe using the most modern user interface available. Works with any card which has a HLLAPI driver, including IRMA, IBM 3270 and Attachmate. Supports DDE, multiple fonts and colors, and comes with a DLL which allows you to write your own applications to access the mainframe. The cost is only $249 per copy.We also carry Netlogic's DDEX - carry on DDE conversations over a network! Have EXCEL access data on a different PC! Only $150 per copy.

VMEM virtual memory system - supports double indirection and automatic swap file compaction. Your applications no longer have to be constrained by 640K!!! $79 with full source code.

MAGMA SOFTWARE SYSTEMS

15 Bodwell Terrace
Millburn, New Jersey 07041
(201) 912 - 0192 (voice)
(201) 912 - 0668 (BBS 2400/1200 baud)



Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from "A Simple Terminal Program . . . ," by W. S. Hall

1

LISTING 1

```
/* program entry point */
int FAR PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
int cmdShow;
{

    MSG msg;

  // initialize program
    if (!InitProgram(hInstance,hPrevInstance, lpszCmdLine, cmdShow))
        return FALSE;

  // stay in this loop until window is destroyed.
    while (TRUE) {
        if (PeekMessage((LPMSG)&msg,NULL,0,0,PM_REMOVE)) {
            if (msg.message == WM_QUIT)
                break;
            TranslateMessage((LPMSG)&msg); // translate keydown/up to chars.
            DispatchMessage((LPMSG)&msg);  // call the appropriate window proc.
        }
        else if (!IsIconic(MWnd.hWnd))      // don't read comm port if iconic.
            ProcessComm();
    }
    return (int)msg.wParam;      // terminate program.
}

/* come here to read the communications buffer if no messages to process */
void NEAR ProcessComm()
{

    COMSTAT ComStatus;
    int num;
    int result = 0;
    int room = BUFSIZE - Buflen;         // check for space

    if ((LineState == IDM_ONLINE) && (!ScrollLock)) {
        if (room > 0) {
            GetCommError(cid, (COMSTAT FAR *)&ComStatus);
            if (num = ComStatus.cbInQue) {       // read number in comm queue.
                num = min(num, room);    // select the smaller value
                if ((result = ReadComm(cid,(LPSTR)(Buffer + Buflen), num)) < 0)
                    result = -result;
                Buflen += result;
                // note, we ignore read errors
                // so result is set to its absolute value.
            }
        }
    }
    if (Buflen)
        PostMessage(MWnd.hWnd, WM_USER, Buflen, 0L);
}
```

LISTING 2

```
/* Main window procedure */
long FAR PASCAL MainWndProc(hWnd,message,wParam,lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
```

2

```
// get our data structure for the window.
PTTYWND pMWnd = (PTTYWND)GetWindowWord(hWnd,0);
PAINTSTRUCT ps;
register int i;

switch(message) {

  // control scrolling with this key.
    case WM_KEYDOWN:
        if (wParam == VK_SCROLL)
            ScrollLock = GetKeyState(VK_SCROLL) & 1;
        break;

  // process menu command.
    case WM_COMMAND:
        WndCommand(hWnd, wParam, lParam);
        break;

/*
   When a character is received, and send it to tty display procedure.
   This is the action when terminal is off-line
*/
    case WM_CHAR:
        HideCaret(hWnd);
      // the loop is a crude way to handle repeat characters
        for (i = 0; i < LOWORD(lParam); i++)
            TTYDisplay(pMWnd, (short)1, (BYTE *)&wParam);
        SetCaretPos(pMWnd->Pos.x, pMWnd->Pos.y);
        ShowCaret(hWnd);
        break;

  // recreate caret and reset its position
    case WM_SIZE:
        pMWnd->Width = LOWORD(lParam);
        pMWnd->Height = HIWORD(lParam);
        pMWnd->Pos.y = pMWnd->Height - pMWnd->CHeight - 1;
        if (hWnd == GetFocus()) {
            CreateCaret(hWnd, (HBITMAP)NULL,2,pMWnd->CHeight);
            SetCaretPos(pMWnd->Pos.x, pMWnd->Pos.y);
            ShowCaret(hWnd);
        }
        break;

  // window has focus, so window can create a caret.
    case WM_SETFOCUS:
        CreateCaret(hWnd, (HBITMAP)NULL,2,pMWnd->CHeight);
        SetCaretPos(pMWnd->Pos.x, pMWnd->Pos.y);
        ShowCaret(hWnd);
        break;

  // window has lost focus, so window must kill the caret.
    case WM_KILLFOCUS:
        HideCaret(hWnd);
        DestroyCaret();
        break;

  // main window now exists so do some initialization.
    case WM_CREATE:
        WndCreate(hWnd,(LPCREATESTRUCT)lParam);
        break;

  // have to close comm port if open before killing window.
    case WM_CLOSE:
        if (cid >= 0) {
            CloseComm(cid);
            cid = INT_MIN;
        }
        DestroyWindow(hWnd);
        break;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

```
    // This application is being destroyed by Windows exec.
    case WM_ENDSESSION:
        if (wParam)         // if true, close the comm port.
            if (cid >= 0) {
                CloseComm(cid);
                cid = INT_MIN;
            }
        break;

    // tell program to exit.
    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    // if iconic, paint the icon window, otherwise redraw text buffer.
    case WM_PAINT:
        BeginPaint(hWnd, (LPPAINTSTRUCT)&ps);
        MainWndPaint(hWnd, (LPPAINTSTRUCT)&ps);
        EndPaint(hWnd, (LPPAINTSTRUCT)&ps);
        break;

    // all other messages go through here.
    default:
        return ((long)DefWindowProc(hWnd,message,wParam,lParam));
    }
    return(0L); /* this means we processed the message */
}

/*
 * Subclass window procedure.  When we are on-line, all window
 * messages come here first
 */
LONG FAR PASCAL MainWndSubProc(hWnd, message, wParam, lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
    PTTYWND pWWnd = (PTTYWND)GetWindowWord(hWnd,0);
    register int i;

    // we are only interested in these messages.
    switch(message) {

        // buffer from comm port has been posted to us.
        case WM_USER:
            HideCaret(hWnd);
            Buflen = TTYDisplay(pWWnd, (short)wParam, Buffer);
            SetCaretPos(pWWnd->Pos.x, pWWnd->Pos.y);
            ShowCaret(hWnd);
            // if anything left over, move it to front.
            if (Buflen)
                memmove(Buffer, Buffer+(int)wParam-Buflen, Buflen);
            break;

        // transmit key from keyboard.  Loop is crude way to handle repeat chars.
        case WM_CHAR:
            for (i = 0; i < LOWORD(lParam); i++)
                WriteComm(cid, (LPSTR)&wParam,1);
            if (!MWnd.LocalEcho)        // if no local echo, then done
                break;

        // all other messages must go thru normal channels.
        default:
            return(CallWindowProc(fpMainWndProc,hWnd,message,wParam,lParam));
    }
    return(0L);
}
```

Code continues.

HARVARD SOFTWORKS

NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

MEET THAT DEADLINE ! ! !

- Combine the raw power of extensible languages with the convenience of carefully implemented functions!
- Execute faster than optimized C!
- Compile 40,000 lines per minute!
- Stay totally interactive, even while compiling!
- Program at any level of abstraction from machine code thru application specific language with equal ease and efficiency!
- Modify routines without recompiling!
- Use source code for 2500 functions!
- Use data structures, control structures, and interface protocols from any other language!
- Implement that borrowed feature, often more efficiently than in its source!
- Use an architecture that supports very small programs or full megabyte ones with a single version!
- Forget chaotic syntax requirements!
- Use subroutine libraries written for other languages! More efficiently!
- Outproduce experienced programmers stuck using conventional languages!

HS/FORTH with FOOPS - The only full multiple inheritance object oriented language under MSDOS!

WAKE UP ! ! !

Forth is no longer a language that tempts programmers with "great expectations", then frustrates them with the need to reinvent simple tools expected in any commercial language.

HS/FORTH Meets Your Needs!

We don't shortchange you with promises. We provide implemented functions to help you complete your application quickly.

Don't judge Forth by public domain products or ones from vendors primarily interested in consulting - they profit from not providing needed tools! Public domain versions are cheap - if your time is worthless. Useful only in learning Forth's basics, they fail at showing its true potential. Not to mention being s-l-o-w.

You can't add extensibility to fossilized compilers. You can easily add features from any other language to **HS/FORTH**. Try our full modifiable multiple inheritance **OOPs** extension or our **Fortran** translator & mathpak!

HS/FORTH has more features and libraries than we can possibly describe here. **Please ask for our brochure!**

HS/FORTH uses MSDOS or PCDOS. Each level includes features of lower ones. Level upgrades: $25. plus price difference. Sources in ordinary text files.

STUDENT LEVEL **$145.**
text & scaled/clipped graphic windows, ellipses, splines, arcs, fills, turtles; powerful parsing, formatting, file and device I/O; shells; interrupt handlers

PERSONAL LEVEL **$245.**
software floating point, trig, transcendental, quad & scaled integer, automatic optimizer for assembly like performance.

PROFESSIONAL LEVEL **$395.**
hardware floating point, turnkey, seal, Rosetta Stone Dynamic Linker, round robin & time slice multitaskers, dynamic string manager, file blocks, sector mapped blocks, assemblers

PRODUCTION LEVEL **$495.**
Metacompiler: DOS/ROM/direct/indirect C data structures & struct compiler, TurboWindow-C graphics library,

Options:

FOOPS+ with multiple inheritance **$ 75.**
286FORTH or **386FORTH** **$295.**
TurboWindow-C (Metagraphics) **$ 99.**
BTRIEVE for HS/FORTH (Novell) **$199.**
ROMULUS HS/FORTH from ROM **$ 95.**
FFORTRAN translator/mathpak **$ 75.**

Number **42** on the Reader Service Card

PRODUCED 2011 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Hall code, continued

LISTING 3

```
/* come here to handle menu items */
void NEAR WndCommand(hWnd, menuitem, param)
HWND hWnd;
WORD menuitem;
LONG param;
{
    HMENU hMenu;
    FARPROC fp;

    switch (menuitem) {
        ...
                    /* other menu items */
        ...
        case IDM_OFFLINE:
            // if currently offline, then change the window proc
            // over to the subclass window procedure.
            SetWindowLong(hWnd, GWL_WNDPROC, (LONG)MainWndSubProc);
            // rewrite the menu bar.
            hMenu = GetMenu(hWnd);
            ChangeMenu(hMenu,IDM_OFFLINE,(LPSTR)szOnLine,IDM_ONLINE,
                        MF_BYCOMMAND | MF_CHANGE);
            DrawMenuBar(hWnd);
            // indicate that we are online.
            LineState = IDM_ONLINE;
            break;

        case IDM_ONLINE:
            // if currently online, reset the window proc.
            SetWindowLong(hWnd, GWL_WNDPROC, (LONG)MainWndProc);
            // redraw the menu.
            hMenu = GetMenu(hWnd);
            ChangeMenu(hMenu,IDM_ONLINE,(LPSTR)szOffLine,IDM_OFFLINE,
                        MF_BYCOMMAND | MF_CHANGE);
            DrawMenuBar(hWnd);
            // show new state.
            LineState = IDM_OFFLINE;
            break;

    }
}

static LOGFONT *pttyfat; // used to find a small font if requested in win.ini

/* called when main window created */
void WndCreate(HWND hWnd, LPCREATESTRUCT pCS)
{
    short width, height, cwidth, cheight;
    TEXTMETRIC TM;
    HDC hDC;
    PTTYWND pWnd;
    FARPROC fp;
    short result;
    HFONT hfont = NULL;
    BOOL smallfont = FALSE;
    int initlen;
    char valstr[80], keystr[20], defstr[20];
    ...
                    /* other code */
    ...
    // Read win.ini to see if the small font should be used.
    // If there is no win.ini entry, make one first.
    initlen = GetProfileString(szAppName, NULL, "", valstr, sizeof(valstr));
    itoa(smallfont, defstr, 10);
    LoadString(hInst, IDS_SMALLFONT, (LPSTR)keystr, sizeof(keystr));
    if (!SearchKey(valstr, keystr, initlen))
        WriteProfileString(szAppName, keystr, defstr);
    smallfont = GetProfileInt(szAppName, keystr, smallfont);
```


MISOSYS

Programmer's Journal downloads are now available from MISOSYS' forum on Compuserve: GO PCS49

Are you still fussing with floppies for BACKUP? Our QIC40 JUMBO tape drive from MISOSYS is your solution!

The Colorado Memory Systems' JUMBO tape drive is one drive that fits all computers. It comes ready for internal use in AT's, XT's, and PC's, and connects to your floppy disk controller. Our adaptor board can be used to connect JUMBO when all FDC connectors are in use. Kits are available for XT/AT or PS/2 machines which convert Jumbo to external use.

- In about 5.5 minutes, JUMBO backs up 10MB's file-by-file - the fastest in the industry! 40MB's gets backed up in about 18 to 20 minutes. Uses industry-standard DC2000 tape cartridges.
- JUMBO plugs into your floppy disk controller to save cost, power, and a slot. Needs 5-1/4" (or 3.5" with faceplate) mounting slot.
- Our tape adapter board mounts in your host computer to provide an additional tape port — especially useful in 286 and 386 computers. When used with our external JUMBO, it lets you share your drive between computers. Note: external adaptor includes "Tape Adaptor"
- JUMBO has custom chips, high speed brushless motors, automatic circuits, no pots to vibrate out of calibration, and fewest parts to assure long life and technical leadership for years to come.

DJ10 Jumbo	$295 ($5S&H)
Tape Adaptor	$75 ($3S&H)
External Adaptor	$110 ($5S&H)
DC2000 tape	$22.50

M/C & VISA. S&H are U.S. only.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS
or 703-450-4181

Now get the clearest view of your disk drive and memory contents with our DED86™

Powerful features in Version 2!

When you need to travel through your disk drive, why settle for a tool that isolates you from the road? DED86 gives you the direct controls you need to explore your disk. It's a full-screen sector-oriented disk/file editor and a page-oriented memory editor. When you want to "unerase" erased files, DED86's flexible KEEP facility does the job without you fussing over FATs.

Check out this list of features

- ✓ Look by cylinder/head/sector, sector or cluster; even reserved/hidden sectors
- ✓ Scan free clusters to search for erased data ✓ Jump about subdirectories
- ✓ Keep sectors & clusters for writing to a file; Great for partial file recovery
- ✓ Edit bytes in hexadecimal or ASCII, zap in 0s; Undo last edit!
- ✓ Search your disk drive, a file, or memory for ASCII or hexadecimal strings
- ✓ Touch a directory file entry with your date and/or time
- ✓ Obtain complete disk statistics in one screen; includes usage data
- ✓ Alter file attributes: archive, system, hidden, read
- ✓ Save/Restore sectors to/from auxiliary buffers; for moving data around
- ✓ DOS subshell available while using DED86 lets you shell to other programs
- ✓ Handles 5.25" & 3.5" drives & RAM disks; even large partition drives

DED86 reduced by more than 50%; Now only $29.95 + $2S&H

Fujitsu Floppy Disk Drives:	5.25" 360K 1/2 height	$75+$4S&H
	3.5" 720K in 5.25" frame	$85+$3S&H
	3.5" 1.44M in 5.25" frame	$99+$3S&H

Number 121 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
// get the system font size
    hDC = GetDC(hWnd);
    GetTextMetrics(hDC, &TM);
    cwidth = TM.tmAveCharWidth;
    cheight = TM.tmHeight + TM.tmExternalLeading;

// If the small font is requested, enumerate all fonts
//  and pick the one with fixed pitch and the smallest height.
    if (smallfont) {
        pttyfat = (LOGFONT *)LocalAlloc(LPTR, sizeof(LOGFONT));
        LoadString(hInst, IDS_FONTFACE, (LPSTR)valstr, sizeof(valstr));
        fp = MakeProcInstance((FARPROC)FindSmallFont, hInst);
        result = EnumFonts(hDC, valstr, fp, (LPSTR)MAKELONG(cwidth, cheight));
        if ((result > 0) && (cheight - result)) {          // found one
            hfont = CreateFontIndirect(pttyfat);           // get the size
            if (hfont) {
                SelectObject(hDC, hfont);
                GetTextMetrics(hDC, &TM);
                cwidth = TM.tmAveCharWidth;
                cheight = TM.tmHeight + TM.tmExternalLeading;
            }
        }
        FreeProcInstance(fp);
        LocalFree((HANDLE)pttyfat);
    }
    ReleaseDC(hWnd, hDC);

    ...
            /* other code */
}

/*
 * this function is called by Windows until all fonts of the specified
 * type have been enumerated.
 */

short FAR PASCAL FindSmallFont(lpLF, lpTM, type, lpData)
LPLOGFONT lpLF;
LPTEXTMETRIC lpTM;
short type;
LPSTR lpData;
{
    register int i;
    static short refheight = 0;
    short refwidth = LOWORD(lpData);
    short height = lpTM->tmHeight + lpTM->tmExternalLeading;
    short width = lpTM->tmAveCharWidth;

// lpData contains the width and height of the system font, which is used
// as a reference value.  On the first pass, it must be initialized.
    if (refheight == 0)
        refheight = HIWORD(lpData);

    // if we find a smaller font, get its parameters.
    if ((height < refheight) && (width <= refwidth)) {
        refheight = height;
        ptyfat->lfHeight = lpLF->lfHeight;
        ptyfat->lfWidth = lpLF->lfWidth;
        ptyfat->lfWeight = lpLF->lfWeight;
        ptyfat->lfCharSet = lpLF->lfCharSet;
        ptyfat->lfOutPrecision = lpLF->lfOutPrecision;
        ptyfat->lfClipPrecision = lpLF->lfClipPrecision;
        ptyfat->lfQuality = lpLF->lfQuality;
        ptyfat->lfPitchAndFamily = lpLF->lfPitchAndFamily;
        for (i = 0; i < LF_FACESIZE; i++)
            ptyfat->lfFaceName[i] = lpLF->lfFaceName[i];
    }
    return refheight;
}
```

Code continues . . .

THE ULTIMATE PROTECTION

StopCopy Plus and StopView:
Quite Simply, the Best Ways to Protect Your Valuable Software Investment.

StopCopy Plus:
Sophisticated Copy Protection that defeats all hardware/software copiers, is loaded with developer options, and remains completely transparent to the honest end user.

StopView:
Advanced copy deterrent that stops hackers, allows tracing of your software, and provides many options without copy protection.

Free demonstration disk available!
Call Today!

BBI Computer Systems
14105 Heritage Lane
Silver Spring, MD 20906
Phone: (301) 871-1094 Fax: (301) 460-7545

Number **303** on the Reader Service Card

Total control with *LMI FORTH*™

For Programming Professionals: an expanding family of compatible, high-performance compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics, floating point, native code generation

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11,34010, V25, RTX-2000
- No license fee or royalty for compiled applications

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

Number **63** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

6

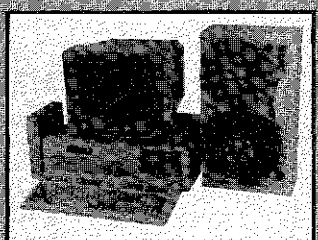
Hall code, continued

LISTING 4

```
/* open a communications port */
static BOOL NEAR OpenAndSetCommPort(hInstance)
HANDLE hInstance;
{
    char modestr[40];
    char commstr[40];
    char errorstr[80];
    char appstr[10];
    int reply;

    // try to open COM1:
    LoadString(hInstance, IDS_FIRSTCOM, (LPSTR)commstr, sizeof(commstr));
    if ((cid = OpenComm((LPSTR)commstr,INQUESIZE,OUTQUESIZE)) < 0) {
        // we failed, so post a message box.
        LoadString(hInstance, IDS_CANNOTOPENFIRSTPORT, (LPSTR)errorstr,
                   sizeof(errorstr));
        reply = MessageBox(MWnd.hWnd, (LPSTR)errorstr, (LPSTR)szAppName,
                           MB_OKCANCEL | MB_ICONQUESTION);
        // if user cancels, quit.
        if (reply == IDCANCEL)
            return FALSE;
        // try to open the next port.
        LoadString(hInstance, IDS_NEXTCOM, (LPSTR)commstr, sizeof(commstr));
        if ((cid = OpenComm((LPSTR)commstr,INQUESIZE,OUTQUESIZE)) < 0) {
            // complete failure, so quit program
            ShowMessage(MWnd.hWnd, IDS_CANNOTOPENANYPORT);
            return FALSE;
        }
    }
    // we opened a port, so we will attach the port name to the title.
    GetWindowText(MWnd.hWnd, (LPSTR)modestr, sizeof(modestr));
    strcat(modestr, commstr);
    SetWindowText(MWnd.hWnd, (LPSTR)modestr);

    // now read in the device control block.
    if (GetCommState(cid, (DCB FAR *)&CommData) >= 0) {
        // read win.ini to find out how to set the port.
        LoadString(hInstance, IDS_PORTSECTION, (LPSTR)appstr, sizeof(appstr));
        GetProfileString((LPSTR)appstr, (LPSTR)commstr, (LPSTR)NULL,
                         (LPSTR)modestr, sizeof(modestr));
        // concatenate the comm data to the name of the port.
        strcat(commstr, modestr);
        // build the DCB according to commstr.
        if (BuildCommDCB((LPSTR)commstr, (DCB FAR *)&CommData) >= 0) {
            // modify some additional DCB fields.
            CommData.XonLim = INQUESIZE / 8;    // Low water mark
            CommData.XoffLim = INQUESIZE / 8;   // High water mark
            CommData.fNull = TRUE;              // ignore null data
            CommData.XonChar = XON;             // use ^Q for xon
            CommData.XoffChar = XOFF;           // use ^S for xoff
            CommData.fOutX = TRUE;              // set outbound flow control
            CommData.fInX = TRUE;               // set inbound flow control
            // finally, set the values.
            if (SetCommState ((DCB FAR *)&CommData) >= 0)
                return TRUE;
        }
    }
    // everything failed, close the comm port and quit.
    ShowMessage(MWnd.hWnd, IDS_COMMSETTERROR);
    CloseComm(cid);
    return FALSE;
}
```


Desktop 9-Track Tape Subsystem

Now, 9-track tape lets your micro exchange data with minis and mainframes



Simply exchange data files on a reel of 9-track tape.

9-TRACK is the first choice for file interchange among data processing professionals. Now, Qualstar's low cost 1/2-inch 9-track Streaming tape systems bring full ANSI data interchange to IBM PCs or Macintosh giving your micro the freedom to exchange data files with nearly any mainframe or minicomputer in the world.

Available in both 7" and 10-1/2" versions, compact Qualstar tape drives can sit on your desktop, *using less space than an ordinary sheet of paper.* Systems include DOS or XENIX compatible software, coupler card and cables. High reliability 1600 or 6250 BPI capability may be used for disk backup as well as data interchange. Discover the big advantage 9-track tape has over other micro/mainframe links.

DEC HP ALTOS RACR — IBM MAINFRAMES — AT&T UNISYS APOLLO SUN

Call us today!

FOR DETAILS AND TO ORDER
FAX (818) 882-4081
PHONE (818) 882-5822

#1 Selling 9-Track Systems on the Desktop

QUALSTAR

9621 Irondale Ave., Chatsworth, CA 91311

© 1989 Qualstar Corp. All product and company names and trademarks are the exclusive property of their respective owners.

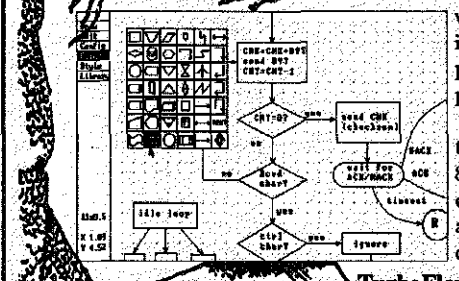
Number 97 on the Reader Service Card

It isn't just another Fable

TurboFlow is serious flowcharting software at a price that won't cost you your kingdom. It's easy to use and more powerful than Merlin's magic.

TurboFlow runs on an IBM PC and features a complete set of ANSI symbols, is menu driven, interfaces withdesktop publishing software, and supports a variety of printers and plotters.

So stop living in the dark ages. Call 1-800-882-5822 and order your copy, or ask for our free brochure.



TurboFlow & Logitech serial Mouse $89

TurboFlow $69

Legendary Software from:

Daytron Electronics, Inc.

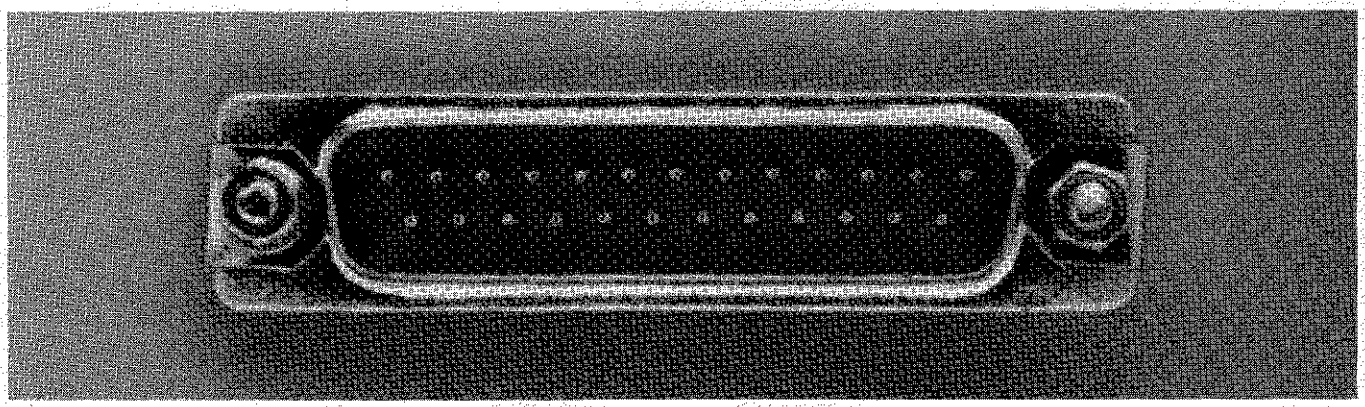
(214)669-2137

610 S. Sherman, Suite 104, Richardson, Tx 75081

Number 308 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

We Know The Ins and Outs of C Communications



South Mountain Software

Programming communications is a time-consuming, complicated, and sometimes tedious programming job. The task of creating a solid communications application can contain more surprises than a magic show.

Professional C programmers have come to rely on South Mountain for fast, safe and efficient communications functions. With a function library as clean and complete as this one, you don't need to know assembler to have complete RS-232 control.

Functions too Numerous to Mention

We can't list all the many functions here, but our library is designed to support the length and breadth of your application's need. From numerous file transfer protocols to intelligent modem control functions, we do it all. If you don't see what you need in the features box below, call us at **800-451-6174**. We probably have what you're looking for.

Freebies

As an added incentive and learning tool, we include the full source code for a BBS system and a terminal package. Both can be incorporated into projects or dissected for programming hints.

We have also designed and included a sub-system for background file transfers. This procedure can save you hours of experimentation and frustration.

BreakOut II Interactive Communications Debugger

For serious communications application development, BreakOut II will save you hours of debugging time. BreakOut II on a PC will allow you to interact with your comm application on a real-time basis, capturing and transmitting data. In the monitor mode (along with a special cabling option) you can monitor the comm session between any two devices on an RS-232 line.

You will be able to spot hardware malfunctions and view line signals up to 19,200 Baud. All of this for hundreds of dollars less than traditional hardware methods.

price $125.

Real People, Realistic Policies

Our tech support and sales lines are peopled by real C programmers. They are able and willing to answer your questions before and after your purchase.

We offer 30-day, money-back guarantees on all our products, and never charge a royalty or run-time fee. If you've forgotten what it's like to be treated as an important customer, give us a call today. You'll like the change.

Features

- **Interrupt-driven transmit and receive to 115,200 Baud**
- **Simultaneous operation on up to 34 serial ports**
- **Intelligent modem control functions**
- **XMODEM, YMODEM, YMODEM-g, and KERMIT**
- **XON/XOFF and hardware flow control**
- **Concurrent file transfer sessions in foreground/background**
- **Timer and CTRL-Break handling**
- **Extensive multi-port board support**
- **Both Microsoft C and Turbo C support included**

Other Products

C-Utility, Essential Graphics, Essential B-Tree, ScreenStar, /*resident_C*/, Guido, Hold Everything

Price $249 inc. Source

No royalties, 30-Day Money-Back Guarantee

1-800-451-6174

SOUTH MOUNTAIN SOFTWARE™

South Mountain Software Inc.

76 S. Orange Ave., S. Orange, NJ 07079

NJ 201-762-6965 Fax 201-762-0118

Essential Graphics, Essential Communications, Essential B-tree
C-Utility Library, Hold Everything, Breakout 2
GUIDO, /* resident_C*/ Screen Star
are trademarks of South Mountain Software Inc.

Number **10** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Human Interfaces with Curses

by Vaughn Vernon

If the issue of UNIX portability for your software has not yet taken on real meaning in your life, it soon will—if it doesn't, you may be sorry. If you end up successful without UNIX, you are special and unique! This article will introduce you to a standard UNIX tool for developing an application human interface. Although you may choose not to use Curses, this will at least make you aware of the issues you will face and how you might deal with them.

Curses does not deal with graphics—it deals with text-mode displays. This is not a problem for most applications, because the majority of applications do not need graphics. In fact, when most people think of graphics, they think of pull-down menus, scroll bars, and dialog boxes. However, most of these "gadgets" can be emulated in text mode, without the overhead of graphics. I'm almost positive that most people have never thought of that. My Formation product has supported many graphics-style-interface gadgets in text mode for about two years now.

If your program depends on graphics mode, you will not be able to depend on portability to UNIX. For instance, although Motif is to be the Microsoft Windows look-and-feel relative for UNIX, the Motif API is not compatible with the Windows or Presentation Manager API—in fact, it is very different. The entire graphics industry is still experiencing much unrest with standards, especially when it comes to portability. As far as relatives are concerned, most standards are like redheaded stepsisters to one another.

What Is Curses?

Curses is a library for C programmers that manages the screen, windows on the screen, and keyboards. The Curses library, completed in mid 1978, was originally designed to give C programmers the ultimate in device independence for character terminals under the UNIX operating system. Additionally, speed of display output and redundant-data suppression were goals that drove the original design and creation of the library.

Curses was written by Ken Arnold at the University of California, Berkeley. He "lifted" intact the TERMCAP database and optimization routines from Bill Joy's VI editor. Later, Mark Horton, also a student at UC–Berkeley, joined AT&T and modified Curses for UNIX System V. Mark Horton added TERMINFO support, along with many other enhancements. Since then, Tony Hansen has added color support with UNIX System V.3.1, and Phong Vo has added several extension routines as of System V.3.2. But perhaps Phong should be remembered for what he *took away* from Curses: He claims to have reduced screen-refresh overhead by some 40 percent!

Device Independence

As you may know, different hardware vendors usually have different physical interfaces to control their hardware. Computer displays and terminals are no different. Therefore, if you attempted to write a piece of software to run on 100 different terminals, you might have to write the software 100 different times. Clearly, this is not a resource-efficient method.

However, a problem even this big can be solved via device independence. You could begin by clearly defining a set of capabilities common to each of the different terminals and put them in a database. Then, you could write a simple library interface to use those common capabilities. The library would deal with the abstract concept of generic screens rather than with physical devices. That is device independence . . . and that is exactly what Curses accomplishes.

Curses is the library interface. The database you will use depends on the system you have: TERMCAP for Berkeley 4.2 BSD and TERMINFO for AT&T System V. These databases contain information about the size of the display area and how to clear the screen, move the cursor, and turn on various video attributes. The major difference between TERMCAP and TERMINFO is content: TERMINFO defines many, many more terminal capabilities than TERMCAP. However, these capabilities are defined only if the given terminal supports them—*TERMINFO is only as good as your hardware.*

Screen Optimization

The topic of optimization is just as important as device independence if you are concerned about the performance of your displayed output. Optimization comes in

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

pretty handy on terminal connections that run at 9600 bits per second or slower. But optimization also works for faster hardware, such as memory-mapped displays. Curses' optimization reduces the amount of physical data that goes to the display to only what absolutely must be sent. That sentence says a lot, but how is it done?

Curses keeps memory buffers that represent the way your application screen and windows should look. When you request that a string or character be written to a window, Curses causes physical output to be suppressed and simply records what portions of the window have been changed.

After making changes to the window and placing the window's cursor, you call the window-refreshing function. At that time, the changed portions of the window buffer are overlaid onto the screen buffer, and the cursor is mapped to screen coordinates. Like the window, the screen buffer is marked for its changed areas before physical output is performed. After some involved computations, Curses sends only the characters from the screen buffer and the control commands from TERMCAP or TERMINFO that are necessary to make the physical display look the way you want it to look.

Consider an instance when optimization is extremely important: You have a screen full of text and you decide to scroll the full screen of text up one line. What will Curses do—redraw the entire display? If you think about the problem, the same data is on the display as before except for only two lines: the top and the bottom. Why should thousands of characters be sent to the display rather than only the few dozen that are really different?

Thousands of characters shouldn't be sent, of course; and, believe it or not, Curses doesn't send them if it can help it—that is, if the proper capabilities are available with the terminal. It figures out that the display needs to be scrolled, and it scrolls the appropriate lines and merely redraws the bottom line!

Benefits of Using Curses

You could compare using Curses instead of a direct display interface to using C rather than assembly language. Many of you using C do so for a very good reason: Even if you are not interested in C's portability, C is simply easier to use than assembly language and almost as fast. You can even get at the bit level with C! Likewise, even if you are not concerned about portability, device independence, or optimization, the abstract concepts found in Curses are easier and more convenient to use than direct hardware interfaces. Curses is plenty fast, too.

Curses' Underlying Concepts

Since Curses is a library that provides the C programmer with an abstract view of the world of windows, displays, and keyboards, Curses uses some abstract terms and concepts. Getting familiar with these abstract terms and the use of their concepts will help you to write better programs sooner.

Why should thousands of characters be sent to the display rather than only the few dozen that are really different?

The lowest-level concept you are likely to be exposed to in Curses is the video cell. The video cell in Curses has a special data type, called chtype. Variables of type chtype are displayable on the screen. In fact, many video cells make up the Curses virtual screen and its windows. A chtype cell holds both a character to display and the attribute or color with which to display it.

A window buffer represents a rectangular area on the display that is no larger than the physical display and no smaller than one character cell. The window can be placed anywhere on the physical display, as long as it will fit where you attempt to put it.

A window, like a video cell, has a special data type: WINDOW. A WINDOW is a structure that contains variables and arrays. The variables hold values such as the current cursor position within the window, the current default video attribute or color of the window, and some

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

flags that define particular modes of operation. One of the arrays is a two-dimensional array of chtype cells that holds the contents of the window—the way it looks. The other arrays are for optimization; they hold information about what portions of the window, if any, have changed since the last window refresh—the last time it was output to the terminal/display.

A window's current cursor position not only tells Curses where you want the cursor to be located on the display, but it also defines where the operation focus is for the window. In other words, just about all operations on a window—adding or deleting characters or lines—take place relative to the current cursor position.

New data that is written to the window replaces what was previously at the same location. When you change a portion of a window or the entire window, Curses suppresses output until you explicitly refresh the window. While output is suppressed, Curses keeps track of window changes in the window's optimization arrays and keeps the data itself in the window chtype cell array. When you refresh the window to initiate screen output, the changed portions of the window are displayed and Curses marks the window as being completely unchanged (although the window still contains the data you have most recently placed into it).

When you initialize Curses, it automatically creates two windows for you: the standard screen and the current screen. The standard screen, with the C variable name of *stdscr,* is a full-screen window, or a window with one corresponding video cell for each cell on the physical display. Since *stdscr* is automatically created for you, it is a good idea to make use of it before creating more (possibly unneeded) full-screen windows. The current screen is a special window that is not usually used by the application.

The Curses virtual screen holds the image of what the real screen looks like. When you refresh a window, its image gets copied to the virtual screen. You can overlay window images onto the virtual screen with or without causing physical screen updating. This allows you to completely change the complexion of your application display without the user knowing it. As you might guess, this supports the use of overlapping windows.

Starting with UNIX System V Curses, function keys and keypad keys could be used. As the successor to TERMCAP, TERMINFO added the ability to define special keyboard keys in a device-independent way, and Curses exploits this ability. Now all the programmer has to do is tell each window to use keypad mapping on keyboard input. When a special key gets pressed . . . voila! It is immediately at your disposal.

Using Curses Functions

The first thing you need to do to get a Curses program to work is initialize it. You also need to include the Curses C header file, Curses.h, in your program. Here is a program segment that shows the basics of initialization:

```
#include <Curses.h>

main()
{
      initscr();
      /* ... */
}
```

In addition, you will probably want to add the following lines after your **initscr** call:

```
nonl();
noecho();
raw();
```

These set up special input modes for using the keyboard. The **nonl** call tells Curses not to map carriage return to newline. The **noecho** function says not to auto-echo characters to windows as they are input. Finally, **raw**

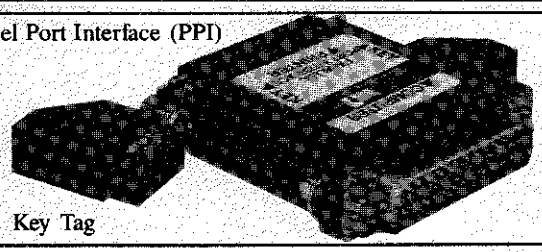
You've seen Michael Abrash — "ON GRAPHICS . . ." Now get the book!

POWER GRAPHICS PROGRAMMING

Order your copy directly from **Programmer's Journal.**

See ad on page 102 for details.

World Class Software Security



The parallel port interface (PPI) connects between the printer port on a PC and the printer cable. The PPI holds two Key Tags, one on each side. Each Key Tag contains a secure custom chip which is pre-programmed by Glenco to only work with the assigned software package. A second Key Tag can be employed to protect another package, or may be used to turn other software packages "on", remotely or on-site.

- STANDARD KEY TAG - Software is protected for an unlimited number of executions. They are pre-programmed to include a sequentially assigned S/N.
- COUPON KEY TAG - Software is valid for a preset number of executions. The Coupon count can be reset remotely or on the customers site by using a second update Key Tag.
- READ/WRITE KEY TAG - With programmable memory. Perfect for companies which have multiple products or a product with several optional modules. By having several packages protected using one Key Tag, your costs are lowered.
- DURATION KEY TAG - Has a clock on board. (Available late '89)

Secure software and data with reliable, effective protection products that won't burden honest users.

Glenco is a world leader in the area of software security products and services. Our copy protection products and data security products are second to none. They are designed to function on a wide variety of third party hardware. We have over 3500 satisfied software firms utilizing our products. We also have a full line of disk based protection systems.

- MACHINES SUPPORTED - IBM PC/XT/AT & PS/2, Macintosh
- OPERATING SYSTEMS - MS-DOS, XENIX, Network, Finder, & Multifinder.
- LANGUAGES/COMPILER - Over 50, including runtime packages, data bases and spread sheets. We have a non-programmers interface as well.

Call or write for more information.

GLENCO ENGINEERING INC.

SERVING THE SOFTWARE INDUSTRY SINCE 1979

721 W. Algonquin Road, Arlington Hts., IL 60005, (312) 364-7638, FAX 364-7698

In Europe contact: SDC Security Systems, The Netherlands
Tel: +31-45-441535, FAX: +31-45-444747

Number **323** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Develop Microsoft® Windows Applications Fast!

"How Did You Get This Windows Application Done So Fast?"
"Everyone Else Is Still Reading the Manuals..."
Two New Products For C And Actor Programmers
WinTrieve™
ISAM indexed file manager. Only $395, no royalties.
Whitewater Resource Toolkit™
Edit dialog boxes, bitmaps, icons and more. Only $195.
Call Now For More Information!

The Windows Advantage

You've selected Microsoft Windows as your graphical user interface. Windows applications are easier for end-users to learn and use. It is **the** standard for PCs.

Now you want Windows applications completed as fast as possible. But it takes time for your programmers to get up to speed on the intricacies of Windows development.

Why The Delay For Windows Applications?

Programming for the Windows environment is very different from character-based programming. Just learning the 450 Windows function calls can take months. Getting used to its event-driven system and memory management takes additional time.

Programmers who are new to Windows development are looking for support to help them produce high-quality Windows programs quickly and efficiently.

3 Ways To Speed Up Windows Application Development

Now there are specific ways to speed up applications development under Windows.

- **Code reduction.** There is a fast, effective way to develop Windows applications. A way that cuts the amount of code needed. A way that by-passes much of the complexity of Windows programming. A way that doesn't compromise performance.
- **Short learning curve.** There is an easy, convenient way to get your entire programming staff up to speed with Windows within weeks.
- **From prototype to complete application.** There is a way to quickly develop prototypes of Windows applications in days, not months. Then, turn those into completed applications quickly.

1 Product, 3 Solutions

There is **one** product which brings **all three** solutions to your Windows programmers. It's called **Actor®**

Actor is a proven product. It's now being used in thousands of software development groups world-wide.

Actor Does Windows

Actor gives your programmers a fast, effective way to develop Windows applications. **Actor's** complete development environment allows you to produce everything from prototypes through stand-alone applications. Programs developed in **Actor** are fast and support all Windows features.

Actor offers interactive source code debugging and testing—greatly reducing development time. Your programmers benefit from high code reusability because **Actor** is an object-oriented programming language. This allows them to build on existing code, eliminating duplication of effort. They can even dynamically link to existing C code.

Actor also provides one more essential: fast, responsive support from The Whitewater Group,® recognized worldwide as leaders in Windows programming.

If you are programming applications for Windows, you should be using **Actor.** Users report that **Actor** cuts Windows programming time by at least 50%.

Act Now!

See for yourself how **Actor** can help your Windows programmers! The first step is to order our free booklet, **For Faster, Better, Smaller, Smarter Windows Programming.** Examine detailed information about using **Actor,** complete with actual source code examples.

Call us at 312/328-3800. We'll send you the booklet and complete details about **Actor.**

Take 60 seconds now to save months of development time in the future.

The Whitewater Group®
600 Davis Street
Evanston, IL 60201
(312) 328-3800

Microsoft is a registered trademark of Microsoft Corporation. Actor is a registered trademark, The Whitewater Group is a registered servicemark, and WinTrieve and Whitewater Resource Toolkit are trademarks of The Whitewater Group, Inc.

Number 73 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tells Curses not to allow the kernel's (the MS-DOS, OS/2, UNIX, or VMS operating system's) device drivers to process or specially treat any keystroke combinations (Ctrl-S, Ctrl-Q, Ctrl-P, etc.) but to send them directly from the keyboard to Curses.

When you are ready to end a Curses program, you *must* call **endwin**. This function restores your display and keyboard to the modes they were in before you called **initscr** and any other setup procedures. If you forget to use **endwin** before you call **exit**, you may be shocked to see the state of the display and keyboard when you return to your command-shell prompt.

To use a window, you always have to create it . . . well, almost always. As you know, *stdscr* is automatically created when you call **initscr**. But let's say that you want to use a smaller, pop-up window. For that you must call **newwin**:

```
WINDOW      *pop;

pop = newwin(10, 50, 2, 2);

if (pop == (WINDOW *)NULL)
     error();
```

The first two arguments to **newwin** are the dimensions of the window—that is, the number of text lines and text columns of window video cells that the window should have. The last two arguments are the physical screen coordinates where the upper left corner of the window should be placed on the display.

If Curses is able to create the window, **newwin** returns to the caller a *pointer* to the newly allocated window. You will use this pointer when referencing the window as long as you need the window. (You reference the window when you add text to it or otherwise manipulate it.) To free the memory used by a Curses window, you call **delwin**.

The functions **waddch** and **waddstr** are used to place text in a window. Since text is added to the window at the current window cursor location, you may want to move the cursor using **wmove** before adding text:

```
wmove(pop, 4, 3);
waddch(pop, 'X');
wmove(pop, 5, 3);
waddstr(pop, "^: Above is the character X.");
```

The **wmove** function moves the cursor to the specified row and column location within the window. In the examples above, the cursor is moved first to row 4, column 3 and then to row 5, column 3.

The **waddch** function adds a video cell to the referenced window. In the above example, the character "X" is placed in the window video cell at row 4, column 3. After the character is placed in the cell referenced by the cursor, the cursor is automatically advanced to the next logical position in the window, and the window is ready to receive the next available character.

The **waddstr** function adds a *NULL-terminated* character array (a C string) to the referenced window. Actually, **waddstr** simply calls **waddch** for each character in the string until the NULL byte is reached. Therefore, all the rules that apply to **waddch** apply to **waddstr**. That makes using **waddstr** convenient. Since the cursor is auto-advanced by **waddch**, **waddstr** supports word wrap from one line to the next.

There are some special routines in Curses that act as a sort of shorthand to calling several procedures. For instance, instead of calling **wmove** and then **waddch** or **waddstr**, you can just code

```
mvwaddch(pop, 4, 3, 'X');
mvwaddstr(pop, 5, 3, "^: Above is the character X.");
```

To get the window image onto the display, you call **wrefresh**:

```
mvwaddstr(pop, 1, 1, "Hello, world!");
wrefresh(pop);
```

The **wrefresh** function makes any and all changed portions of the window overlay onto the virtual screen. Then it forces the output of the virtual screen to the physical screen. The **wrefresh** function itself actually calls two functions to do these jobs: **wnoutrefresh** and **doupdate**. The **wnoutrefresh** function simply overlays a window image onto the virtual screen. The **doupdate** function causes all changed portions of the virtual screen to be sent to the physical screen. Since these two functions can be called individually, you have ultimate control of overlapping windows.

In addition, you may have to use a function named **touchwin**. This function makes a window appear to Curses as being completely changed, even if it is completely unchanged since its last refresh. By using **touchwin**, you force **wnoutrefresh** to overlay the window image onto the virtual-screen image.

The short example in listings 1 and 2 shows how you can use a window push-and-pop method to zap windows on and off the display. This example maintains a stack of window pointers in the order of overlap. When you want to display a window, you call **winPush**. When you want to hide the most recently pushed window, you call **winPop**.

Another tip is to always have a full-screen window, such as *stdscr*, as the bottom-level window. This ensures that nonoverlapping regions of the window you are removing will be erased, too. So, the first call to **winPush** above would be

```
winPush(stdscr); /* bottom window */
```

When text is added to a window, it is added with the current window attribute/color. The current attribute/color is A_NORMAL when the window is first created. To change the current attribute/color, you call **wattrset**:

```
waddstr(pop, "This string is in A_NORMAL.\n");
wattrset(pop, A_REVERSE);
waddstr(pop, "This string is in A_REVERSE.\n");
```

To use monochrome attributes, you directly call **wattrset** with the attribute or bit-ORed attributes you want to set. To use color, you need to first create a color pair:

```
initscr();
start_color();

/* ... */
```

Continued on page 48

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

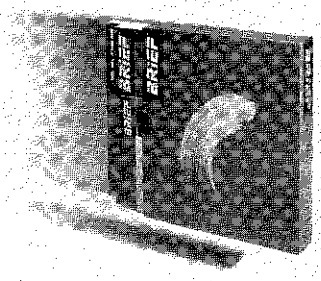
Your operating system finally caught up with our editor.

Introducing BRIEF for OS/2.
It's the most versatile editor available, in a powerful new environment.

Now, you can have the best multi-level undo available, potent search and translate capabilities, and two powerful (but easy-to-use) macro languages. Plus all the inherent advantages of OS/2, like multitasking.

And BRIEF is still easy to use, and easy to learn. Commands are consistent and intuitive, and the documentation is clear and concise. Best of all, it's still affordable—just $195.

If you've made the investment in time and money to move up to OS/2, make sure you're working with an editor as advanced as your operating system.



Call today to order your copy of BRIEF for OS/2. And stop playing catch-up.

Special discount for DOS BRIEF users.

Call 1-800-343-7308.

BRIEF® for OS/2

UnderWare, Inc., 321 Columbus Ave., Boston, MA 02116 (617) 267-9743

Number **309** on the Reader Service Card

Multitasking

for the IBM-PC/XT/AT with

Multidos Plus

Ideal for developing process control, data acquisition, communications, and other applications which require a multitasking environment. Check these features which make Multidos Plus an unbeatable value:

- Run up to 32 programs concurrently
- Your software continues to run under PC-DOS or MS-DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most compiled languages or assembly language.
- Operator commands to load and run programs, change priority, check program status, and to abort/suspend/resume task execution.
- Optional independent foreground and background displays.
- Programmatic interface through INT 15H for the following:
  - * Intertask messaging to send, receive, check or wait on 63 different message queues.
  - * Task control by means of semaphores. Get, release, check, or wait on semaphores.
  - * Change task priority.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.

Requires: IBM PC/XT/AT or true clone. MS-DOS (or PC-DOS) 2.0 or later. Enough memory to hold Multidos Plus (64KB) and all your application programs. You may also need 4 or 16KB of memory for "hidden screens" for each active task.

$24.95 or

$99.95 with source code (written in Microsoft QuickC 2.00 and Assembler)

To order call (800) 678-2141. FAX (508) 655-8860. For technical information and foreign orders call (508) 651-0091. Visa and Mastercard accepted. Or send check or money order (Drawn on U.S. banks only) to:

Nanosoft

13 Westfield Road
Natick, MA 01760

Add $2.00 for shipping & handling.
Foreign orders add $5.00 for shipping & handling.
COD orders add additional $3.00
Massachusetts shipments add 5% sales tax.

Number **109** on the Reader Service Card

PLEASE KEEP IN TOUCH . . .

Just send us your new address along with your previous ZIP code, and use this handy form! Thanks.

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____

ZIP CODE _____ OLD ZIP CODE— _____

TELEPHONE _____

Please mail to:
Programmer's Journal
P. O. Box 30160
Eugene, OR 97403

Or call toll free
800-234-0386

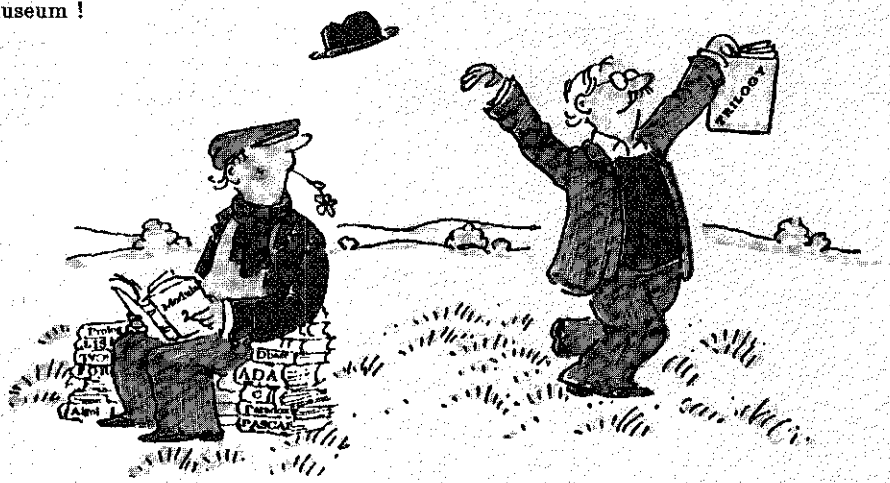
PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HAVE YOU REALIZED THAT

YOUR LANGUAGE IS A DINOSAUR !!!

THE ONE-PARADIGM LANGUAGE YOU ARE USING NOW, such as a third generation procedural language (Pascal, Modula, C, ADA, etc.), a fourth generation database language (Dbase, SQL, etc.) or a fifth generation logic language (Prolog, Lisp), **IS FACING EXTINCTION !** Put it in a museum !

TODAY, programmers around the world use TRILOGY - the language designed to *'create a revolution'* (R.Fischer, PC Welt, Germany)! **TOMORROW you will be one of them, and you will love it!**



TRILOGY introduces the new Multi-Paradigm Generation of computer languages! It seamlessly integrates the last three generations of languages into a single genuine **GENERAL PURPOSE LANGUAGE of unprecedented power** and expressivity, radiating simplicity and elegance. With this *'weapon that dramatically improves your productivity'* (Dr.J.Pournelle, BYTE), you can save 35-40 % of code.

Large-scale integration of software, until now believed to be unmanagable, was made possible by discovering the logic theory applicable to all programming paradigms. We have based **TRILOGY** on this theory, and we have managed to hide all of the difficult concepts behind a **friendly Pascal-like syntax** as easy as "if-then-else".

TRILOGY, an original product of inventive thinking, is the result of a **decade of research** and three years of implementation. We are offering you a *'deluxe experimenter's system'* (Ph. Chapnick, AI Expert) rejecting the unscientific *ad hoc* methods of language design used in all of the one-paradigm languages.

LOOK AT TRILOGY TAKING THE WORLD BY STORM :

"... Trilogy is what Pascal would be if it were designed today with the knowledge of logic and database programming we now have ..." *Ray Weiss, Electronic Engin. Times, Dec. 1987, 'Pascal from Prolog'*

"... The West Coast Computer Fair has long been the spawning ground for important micro products ... It does not happen as often now but it's still possible, as for example Trilogy ... what I've been able to do with (Trilogy) really impressed me ... you may find yourself with a secret weapon that dramatically improves your productivity ... I am not alone in my impression that Trilogy may ... be one of the "big" languages like Pascal or Prolog" *Dr.Jerry Pournelle, Columnist, BYTE, July 1988 column 'Computing At Chaos Manor'.*

"... It's an incredibly fast language ... Everything I saw in this language told me that its author was no lightweight. And it was obvious, even in the manual, that this had been a long term project ... (Trilogy is) too good a language and too good a development environment." *Dave Thompson, Editor-In-Chief, Micro Cornucopia, January 1988*

"... programmers will soon come to expect languages that support several paradigms. One such language already available, Trilogy, can be used for procedural, functional, logical, constraint, and data base programming ... Prolog programs, using backtracking, typically require several hours to solve this problem (Zebra). Trilogy, using more powerful (built-in constraint satisfaction) decision procedures, finds the answer in less than one second." *Dr. Peter Gregory, the author of 'Programming in Pascal', Computer Language, April 1988.*

"... Trilogy embodies a wealth of features that ranges from the unusual to the unheard of ... we expect to see Prolog-Trilogy religious wars break out ... If we had to enlist, we'd sign up with Trilogy." *Mickey Williamson, the author of 'Artificial Intelligence for Microcomputers', PC Week, February 2, 1988*

"... Trilogy is a new programming language which integrates the knowledge acquired in computing science in the last decade. The result is astonishing. In the competition for the most economic programming system, (Trilogy) ... has a good chance to win the first place ... When Trilogy gets the attention it deserves, it will create a revolution ..." *Ronald Fischer, PC WELT, Germany, November 1988, 'Pascal und Prolog in Einem'.*

"... Trilogy is a unique product. It is the only product I have seen that combines logic programming with procedural and database concepts." *Alex Lane, Contributing Editor, BYTE, March 1988*

"... Trilogy ... is a deluxe experimenter's system." *Philip Chapnick, AI Expert, May 1988, 'Don't Avoid Conventions - Be Multilingual'.*

"... The three levels of Trilogy offer the strength of conventional procedural programming languages, database languages, and declarative languages used in artificial intelligence ... (Trilogy) is much more simple than expected ... people with experience in programming ... will be able to learn it very fast." *Computerworld Switzerland, March 14, 1988, 'Die Trilogie der Sprachen'.*

"... Trilogy, the new programming language from Canada that is receiving attention in the U.S. AI community, is adding an MS-DOS module and a Graphics capability (module) ... (Trilogy) is modular and includes a full screen editor, module librarian, interactive compiler, on-line linker and loader, cross-referenced help, and automatic make. Windowing and external file access are supported, as are 80x87 coprocessors. The Pascal-like syntax (fits into one page) ..." *AI Expert, October 1988, 'Trilogy keeps growing'.*

TRILOGY runs on IBM PC, XT, AT, and compatibles, under DOS.
Order from:

Complete Logic Systems
741 Blueridge Ave.
North Vancouver
B.C., V7R 2J5, Canada.
Telephone: (604) 986-3234

VISA, MC, money order welcome. Add US $ 7 for shipp./handl. in North America, or US $ 13 for oversees shipments.

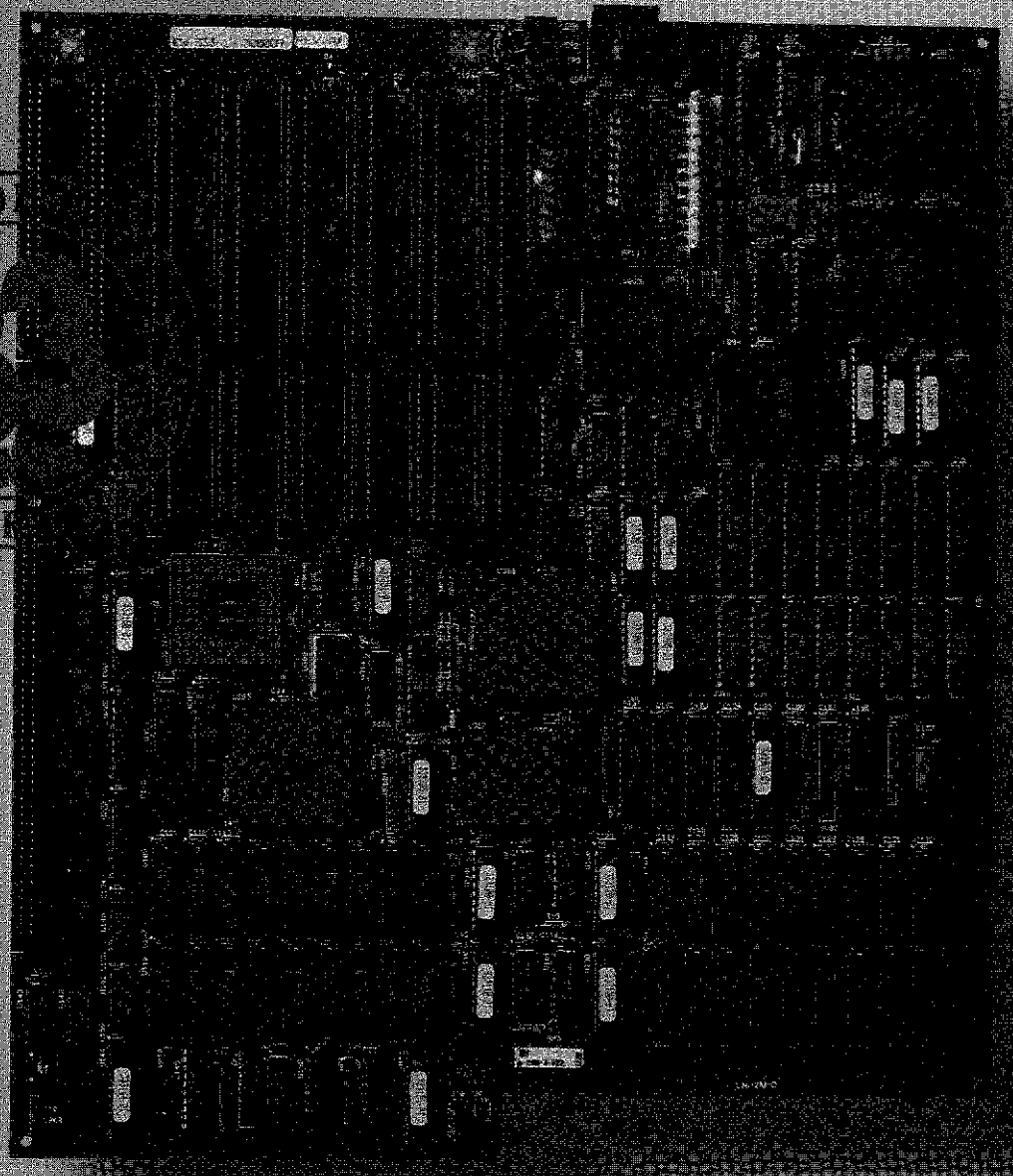
TRILOGY ADVANCED PACKAGE: US $200.00
(package includes: TRILOGY V1.3, TOOLBOX, LINKER)
or you can order separately:

TRILOGY V 1.3: US $100.00 (Compiler & Environment)
TRILOGY TOOLBOX: US $ 50.00 (DOS connect., Graphics)
TRILOGY LINKER: US $ 50.00 (Creating .EXE files)
2 Demo disks: US $ 10.00 (Running Mini-Trilogy)

70% discount offered to all registered users of TRILOGY V1.3 on **TRILOGY++** with OOP capabilities to be released in 1990.

Number 301 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



IT Hz SO GOOD!

If your computer feels slow, we know where it hertz. For a fast cure, get our new 386 MotherBoard/33Mflz. It will accelerate your PC, PC/XT or PC/AT to speeds equal to the IBM Model 80. And faster. We've built in 4 Megabytes of high speed RAM, 64K of RAM cache, and a 387 math coprocessor socket for blistering performance!

Network Savvy. With the 386 MotherBoard/33MHz, your computer becomes a file server or workstation that makes Novell networks *scream*. Beyond speed, you'll enjoy compatibility with Token Ring, Ethernet, Arcnet, and most other network cards.

The UNIX Engine. Great for VARS, Systems Integrators and UNIX OEMs, the Hauppauge 386 MotherBoard/33MHz runs SCO Xenix, Interactive 386/ix and AT&T's UNIX System V. With its PC/AT compatible I/O system, our 33MHz board accommodates the latest in disk control, graphics, and network I/O cards -- just to name a few.

CAD Capability. Ever find yourself growling, "faster, faster!" at your AutoCad and other CAD programs? We've got good news. The 386 MotherBoard/33MHz boosts your math and graphics applications, and supports the high speed 387-33 and Weitek Abacus math coprocessors.

Technical Features The 386 MotherBoard/33Mflz includes: ▪ 4 Megabytes of high speed 32-bit memory, expandable to 64 Megabytes ▪ 64K of 20 nsec cache memory ▪ Six 16-bit expansion slots, one 8-bit and one 8-bit/32-bit slot ▪ PC/AT compatible I/O system and BIOS for support of OS/2 and UNIX.

386 MotherBoard/16MHz for the PC, XT, AT and Compaq portable	$1395
386 MotherBoard/20 and 25MHz for the PC/AT	from $1695
386 MotherBoard/33MHz with 4 MBytes of RAM	$3995
32-bit RAM Boards	from $ 295

Prices subject to change

Why endure a painfully slow computer? With Hauppauge's 33MHz MotherBoard, your computer flz so good! For more information and fast delivery, call: 1 (800) 443-6284. In New York: (516) 434-1600. Available on GSA contract #GS00K-89AGS6353.

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

Trademarks: IBM PC, XT, AT, PC and OS/2 IBM Intel 386 Intel Corp. Windows 386, Microsoft Corp. PC-DOS Compaq

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
if (has_colors()) {
     init_pair(1, COLOR_WHITE, COLOR_BLUE);
     wattrset(pop, COLOR_PAIR(1));
     waddstr(pop, "This string is white-on-blue.");
}
```

You call **start_color** just after **initscr** to initialize the Curses color interface. The **start_color** function sets up some variables for you: COLORS and COLOR_PAIRS. The COLORS variable is an int that tells you the number of available colors for your terminal/display. The COLOR_PAIRS variable is an int that tells you how many color pairs you can build for this terminal/display. If you have a color terminal/display, **has_colors** returns TRUE; otherwise, it returns FALSE.

When you know you have colors available, you can build a color pair. The **init_pair** function takes the arguments *pair number* and *foreground* and *background* colors. The *pair number* argument must be between 1 and COLOR_PAIRS - 1. To set the current color for the window, you again call **wattrset**, but this time with the color pair you have initialized:

```
wattrset(pop, COLOR_PAIR(1));
```

The COLOR_PAIR macro returns the color pair related to its *pair number* argument.

Along with passing the color (referenced with COLOR_PAIR), you may combine other regular attributes that make sense:

```
wattrset(pop, COLOR_PAIR(1) | A_BOLD | A_BLINK);
```

For instance, turning on A_BOLD causes the foreground color to become a lighter shade than normal and bold black to become gray. You may also want an "alert" message to blink at the user.

You use the **wgetch** function to get a keystroke from the user. If you are interested in getting special keys auto-translated for you, you should also use the **keypad** function. Call **keypad** just after you have created each window:

```
WINDOW      *pop;
int  key;
pop = newwin(10, 50, 2, 2);
keypad(pop, 1);
box(pop, 0, 0);
mvwaddstr(pop, 1, 1, "Press F1: ");
wrefresh(pop);
key = wgetch(pop);

switch (key) {
case KEY_F(1): /* ... */
case KEY_UP: /* ... */
case KEY_DOWN:       /* ... */
case KEY_LEFT:       /* ... */
case KEY_RIGHT:/* ... */
}
```

A value of 1 as the second parameter to **keypad** tells Curses to translate keys when they are input via *pop*. When F1 is pressed, *key* will be set to the constant generated by the macro KEY_F(1). Likewise, other special keys are returned as symbolic constants.

There's More to It

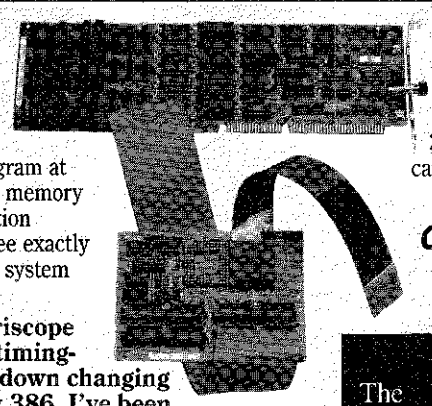
Curses provides a ream of other functions and capabilities, but space does not permit us to indulge; however, I am allowed to whet your appetite! You can use subwindows, pads (large virtual windows), and subpads. You can use line drawing and bordered windows with line-drawing characters. Curses also supports a set of Soft-Label-Keys.

Vaughn Vernon is president of Aspen Scientific and author of Curses/PC (for DOS and OS/2) and Curses/VMS (for VAX/VMS). He has written other development tools to use with Curses, including FAST and FORMATION.

Code follows.

NEW FROM PERISCOPE® Real-Time Debugging For 286s and 386s up to 25MHz

Periscope® IV brings you a new generation of hardware-assisted debugging. It works on virtually any 286 or 386 with an AT-style bus running up to 25MHz with zero wait states. With Periscope IV, you can run your program at full speed while tracking down unwanted memory overwrites. And you can use the information captured in its real-time trace buffer to see exactly what the system is doing, and to improve system performance.



"The hardware really makes Periscope shine, especially when you've got timing-related problems. I can now track down changing pointers and altered buffers on my 386. I've been using it to debug Crosstalk® Mk. 4 and there's just no better way to do it," says user Jeff Garbers, Crosstalk Communications' Director of Software Development.

Call now for our FREE four-page Model IV brochure or to talk about your debugging needs. Prices range from $145 for just software to $2,995 for a Model IV for a 25MHz 80386 system. We accept major credit cards and qualified company purchase orders.

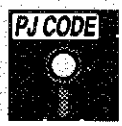
Order your Periscope, Toll-Free, Today! 800-722-7006

The Periscope Company, Inc.

1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080
FAX 404/872-1973

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from "Human Interfaces . . . ," by Vaughn Vernon

LISTING 1

```
/*
 * winstack.h           support winPush(win) and winPop().
 *
 * (c) Copyright 1988 Aspen Scientific
 * All Rights Reserved.
 */

#define WIN_MAX         32      /* this should be enough */

extern int winPush( WINDOW * );
extern void winPop( void );
```

LISTING 2

```
/*
 * winstack.c   winPush(win) and winPop().
 *
 *              manage a stack of CURSES windows.
 *
 * (c) Copyright 1988 Aspen Scientific
 * All Rights Reserved.
 */

#include <curses.h>
#include "winstack.h"   /* some defines */

/* the stack
 */
static WINDOW *ws[ WIN_MAX ];

static int ws_idx = 0;

int
winPush( win )
WINDOW *win;
{
        /** stack overflow?
         **/
        if (ws_idx == WIN_MAX)
                return (0);             /** error **/

        touchwin( win );
        wrefresh( win );

        ws[ ws_idx++ ] = win;

        return (1);                     /** good **/
}

void
winPop()
{
        register int i;

        /** stack underflow?
         **/
        if (ws_idx == 0)
                return;

        ws[ --ws_idx ] = (WINDOW *)0;

        i=0;
        while (i < ws_idx) {
                touchwin(ws[i]);
                wnoutrefresh(ws[i]);    /* to virtual screen */
                ++i;
        }

        doupdate();     /* to physical screen */
}
```


You've seen Michael Abrash — "ON GRAPHICS . . ." Now get the book!

POWER GRAPHICS PROGRAMMING

Order your copy directly from *Programmer's Journal.*

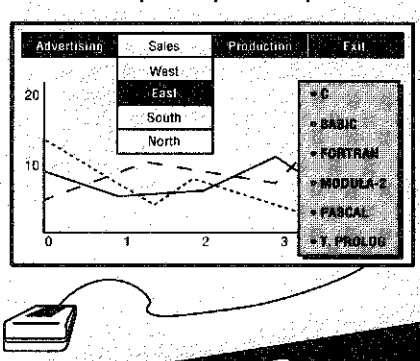
See ad on page 102 for details.

IF YOU PROGRAM, PLEASE TRY

HI-SCREEN XL™

WINDOWS - MENUS DATA ENTRY - HELP

- **WYSIWYG** screen editor • draw • paint • copy • merge • undo • on-line ASCII table
- **WINDOWS** • overlapping • scrollable • for menus, data entry, help screens
- **MENUS** • pop-up • pull-down • scrolling • Lotus-style • "your-own-style"
- **HELP** • context-sensitive • at screen and field level • fully automated*
- **DATA VALIDATION** • type • range • format • color • justification • case
- **INPUT MANAGEMENT** • field-by-field and full-screen input • user-defined exit & function keys • arrays • hot keys



WITH MOUSE AND GRAPHICS SUPPORT*

- Use the **mouse** to select, scroll, validate, move cursor, click in dialog boxes, etc.
- Mix **graphics and text** screens in your programs. Capture any CGA, EGA or VGA graphic images and combine them with menus, windows, dialog boxes, etc.

Call toll-free: **1-800-338-2852**
in CA: 415-397-4666 • Fax: 415-397-4677

HI-SCREEN XL $149
HI-SCREEN XL *Professional Series* $325

30 day money-back guarantee • No royalties
* Professional Series version only

For C, BASIC, FORTRAN COBOL, dBASE, PASCAL...

Softway, Inc.
500 Sutter #222, San Francisco, CA 94102

Number 110 on the Reader Service Card

PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Cooking with Stereo Video

by Guy Cole

How many times have you wished for a larger display?

It's amazing how quickly an 80-character by 25-line display fills up. Oh, certainly, with an EGA or VGA one can go to 43-line or 50-line displays, but how long before eyestrain sets in? Or you can build windows into oblivion, but the finished product can look cluttered—not to mention the memory a window-intensive application requires. Memory can be expanded, and larger disk drives are always available, but the size of the display seems to be a constant. Given that the worth of a program is typically judged by its display, an uncluttered, thoughtful screen layout is absolutely essential.

What about two displays? Although it's probably not the equivalent of doubling the video area, it is certainly an improvement. Last year, I was confronted with a project that required a large amount of text keyed to geographic displays. My solution was to draw the maps on an EGA and place most of the amplifying data on a monochrome display. This arrangement provided greater flexibility with the display templates, thanks to the increased display area, and it was inexpensive to implement.

There are other benefits to a system sporting two video adapters. CodeView, Periscope, Turbo Debugger and the Turbo Integrated Environments for both Turbo C and Turbo Pascal support dual-monitor options. Imagine the ability to execute a graphics program on one dis-

PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

play and watch variables on the other—no flipping between displays, no loss of continuity. It is an excellent productivity enhancement, creating something like an inexpensive workstation environment. And best of all, it can probably be done on your current PC.

This article explains how running two video displays is possible and how to implement this setup on your machine in Turbo Pascal. The Pascal module is called AltMon, for *Alt*ernate *Mon*itor. AltMon is completely self contained, imitates all Turbo Pascal video text functions, and is ready to be employed in your own applications. An additional program, AltTest, demonstrates methods for employing a dual-monitor setup. *[NOTE: Because of space limitations, AltTest is not printed in this issue. It is available on the Code Disk for this issue as well as through our bulletin board services (see pages 54 and 96). —Ed.]*

No Conflicts Allowed

Video adapters on the PC are memory-mapped devices. A video card of a specific type has a reserved space in memory in which to operate. Monochrome memory begins at B000:0000h, and CGA adapters start at B800:0000h. EGA and VGA cards have several possible memory addresses, depending on the video mode. These addresses translate to the upper left corner of their respective displays. Video RAM in text mode is arranged in character/attribute pairs. That is, each character has an associated video attribute, which will specify the foreground and background color of the character and whether or not the character should blink. The arrangement and contents of the character/attribute pairs remains the same; only the starting address is dependent on the adapter type.

The monochrome adapter is lower in memory than the color display. Starting at B000h, a monochrome adapter requires space for 80 characters by 25 lines, multiplied by 2 (one byte for the character, and one for the attribute)—4,000 bytes of RAM. Note that the end of the monochrome area is below B800h, where the CGA takes over, and we can infer that no memory conflict occurs between the two. (A memory conflict occurs when two devices compete for the same location within the system.) The result is that a monochrome and CGA adapter can peacefully coexist within the standard PC memory map.

The I/O ports that control the video adapters also conveniently work out to allow a dual-monitor arrangement. The monochrome ports are in the range 3B0h to 3BFh, while color ports can be found at 3D0h to 3DFh. Again, there is no conflict between devices.

Installation and Configuration

Within the PC world, everything has its subtle and not-so-subtle variations. For the monochrome adapter, the not-so-subtle must certainly be the popular Hercules-graphics-card standard. In graphics mode, most Hercules cards have two video pages, of which the second page encroaches upon the CGA portion of the memory map. This is to be avoided: If you are shopping for a monochrome card, be sure that you can disable the Hercules graphics option with a hardware jumper or software port on the card. Some commercial packages will test for the presence of a Hercules-type card and then involuntarily place you in graphics mode. The video cards will get surly at each other, and your system will probably crash. Since we are talking about a CGA, EGA, or VGA on the bus, along with the monochrome, just let them handle the graphics work, and disable the graphics option on the monochrome side.

Some enhanced graphic adapters need to be configured for dual-monitor operation—consult the installation instructions for your particular model. My EGA card requires a different DIP switch configuration depending on whether or not it is the boot monitor and whether it is to come up as a monochrome or color display.

If the candidate machine is an AT, it's very likely the configuration will also need to be changed. My AT clone uses Phoenix BIOS version 3.02 and needed to have the primary monitor set to "other or none"—strange option, but it works, and the machine seems satisfied.

Imagine the ability to execute a graphics program on one display and watch variables on the other—no flipping between displays, no loss of continuity.

Let's assume you've robbed your old XT of its monochrome video card and display. Your AT lies with its cover off on the kitchen table. The EGA or VGA card and BIOS have been configured. Picking a spare slot, confidently insert the monochrome card, connect the monitor, and turn the machine on. (A confident manner is important: Computers, like animals, can sense fear.) After you get to watch the usual ROM commercial, memory

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

diagnostics, and whatever AUTOEXEC.BAT performs, the prompt appears. You're now cooking on stereo video.

3-D Glasses Not Required

Well, maybe not quite stereo video, since each monitor is a separate entity and must be addressed as such. MS-DOS supports only one video adapter at any time. Although two displays are present, only one will be "known" to the system. From the command line (or batch files), you can switch the active display using the DOS MODE command (MODE MONO for the monochrome display and MODE CO80 for the color display).

Some words about commercial software and dual monitors are in order at this point. Terminate-and-stay-resident (TSR) programs have a tendency to remember the video mode that was current when they were loaded. Older versions of Sidekick or Turbo Lightning certainly fit this category. These need to be configured with colors that are visible on both display types, or you won't always be able to determine what option is highlighted. Some experimentation will yield an effective compromise color palette. Usually, non-TSR programs make some effort to determine the current display type when started and adjust themselves. BRIEF is a good example—it features different configuration files for both mono and color modes.

A confident manner is important: Computers, like animals, can sense fear.

Stereo Video in Turbo Pascal

Now it's time to consider employing two monitors in your own programs. Probably the simplest way to give the appearance of driving two monitors is by switching the active monitor during program execution. This method has the advantage of enjoying all the normal facilities for video control, such as clearing the display and setting the cursor position.

Switching active displays cannot be accomplished utilizing BIOS services alone—the BIOS equipment-list variable, located at 0000:0410h, must also be altered. (This is the same information returned by interrupt 11h, "equipment-list service.") Within this word, bits 4 and 5 specify the video equipment detected during system boot. These bits must be updated to reflect the display you wish to become current (10b for color, 11b for monochrome). After the BIOS equipment list has been updated, issue the appropriate mode change via interrupt 10, service 0, and the default monitor should be changed.

Function **Swap_Monitor** illustrates this procedure. Upon entry, it requests the current video mode and reads the current contents of the BIOS equipment list. If the current mode is color, the target mode will be monochrome. An 11b is ORed into the equipment list, which specifies monochrome mode, followed by a mode request to video-mode 7 using interrupt 10h. To switch from monochrome to color, the video-equipment bits are cleared with a mask and a 10b is inserted. Again, building the new equipment-list variable is followed by a video BIOS mode change, in this instance to mode 3. Finally, **Swap_Monitor** returns TRUE if color mode was set or FALSE for monochrome, just in case the calling program needs some feedback.

The disadvantage to using a routine similar to **Swap_Monitor** and the intrinsic video-control routines to maintain dual video adapters is that each time the monitors are switched, the target monitor clears its display. If your information is relatively static (help menus, perhaps), then simply switching the default monitor could be a reasonable approach. But more-dynamic applications demand greater sophistication.

Since the normal video support is reserved for the primary monitor, some other method must be employed. Writing directly to the video RAM on the alternate display

The
Heap Expander™
version 2.05

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

only $79.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:
- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*CA residents add 6 1/2% sales tax. Foreign customers, please add $4.00 for shipping and handling.

The Tool Makers
P. O. Box 2151
Santa Cruz, CA 95063
(408) 458-0690

Number 166 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

seemed the logical option, and in practice it has turned out rather well. It is probably easiest to employ the color display as the default monitor and use the monochrome display as "alternate." In this fashion, you will be able to utilize all the graphics functions you've become accustomed to, without conflict. Also, since text modes have fewer options, coding is an easy task.

A Full-Featured Turbo Pascal Unit

The accompanying Turbo Pascal unit, ALTMON.PAS, imitates all of the intrinsic text functions available from Turbo. The only difference is that functions are prefixed with an "Alt_". For example, to clear the alternate display, simply invoke **Alt_ClrScr**. Your program should call **AltMon_Setup** prior to using any of the services within AltMon.

One of the truly endearing qualities of Turbo Pascal is the ability to specify addresses and manipulate ports. Within ALTMON.PAS, this is used to good effect. **Display_Record** describes the arrangement of the character/attribute pair, which **Line_Type** builds on to represent a display line, followed by **Display_Type** to symbolize an entire display. In the VAR declarations, **Display** is defined to be of **Display_Type**, located at B000h, which maps the entire structure onto the monochrome video RAM. In this fashion, the alternate-monitor memory can be referenced via familiar *X* and *Y* coordinates. Note that if the alternate display needed to be CGA, the only change required would be the address referenced by **Display**: from B000h to B800h.

Let's begin dissection of ALTMON.PAS with **AltMon_Setup**. Normal video attributes are selected, and the logical-window limits are set to the physical-display limits. This is followed by the clearing of the alternate display and the enabling of the pseudocursor. AltMon is now open for business.

Alt_ClrScr performs the clearing of the alternate display. This procedure begins by padding the logical top line with spaces and then copies the cleared top line down for the remainder of the window. Use of block-memory transfers such as MOVE are much faster than byte-by-byte transfers with a loop, and they should be employed at every opportunity. Just prior to exit, the pseudocursor is set to the upper left corner of the current window, just as with the normal ClrScr.

That empty display is just screaming out for something to be written on it. **Alt_Write** and **Alt_WriteIn** will enable you to place information onto the alternate monitor. It is unfortunate that only strings can be passed to these procedures. Pascal, by its very nature, is a strongly typed language. This dictates that the quantity of arguments and their data types be defined at compile time, and so variable argument lists are not possible. I tried some alternatives. One used generic pointers, and the other was a linked list of records specifying the data and format control. In the end, the generous string-handling abilities of Turbo Pascal tipped the balance, and the convention is that all arguments passed to **Alt_Write** or **Alt_WriteIn** will be strings.

Within **Alt_Write**, I first check to determine if scrolling is required. Then each character is taken in turn and tested to ensure that it is printable. If it is not found to be printable, execution branches to service the possibility that it is a control character. All of the legal Turbo Pascal control characters are supported—bell, backspace, newline, and carriage return. In the case of normal characters, the cursor is advanced and then tested to see if the logical end of line has been exceeded. If so, a wrap to the next line is performed, accompanied by another test for scrolling, and the cycle continues for the length of the string.

The pseudocursor position can be requested and updated by the utilization of **Alt_WhereX**, **Alt_WhereY**, and **Alt_GotoXY**. These functions are easily implemented by merely reporting or altering the array indices.

Video attributes are extremely useful tools for delimiting different fields or attracting operator attention. AltMon provides complete support for reading and setting monochrome video attributes. The *TextAttr* variable is duplicated with **Alt_TextAttr**, with one important

DOS & OS/2
Free Demo Diskette Available

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

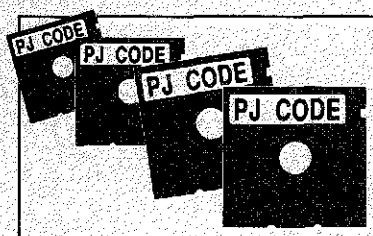
(Outside U.S., add $5 for shipping and handling)

800-344-4096
(In NJ, 609-779-0784)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
496 N. Kings HWY Suite 130
Cherry Hill, NJ 08034

Number 158 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PJ Code Is Available!

On disk from:

Programmer's Journal
P. O. Box 30160
Eugene, OR 97403
800-234-0386
503-747-0800 in Oregon
All Code Disks $10.00 each.
To order, see ad on page 96.
PJ Code DiskScriptions
are available for only $39.95.
To order, see ad on page 8.

On BBS from:

EDI
1332 Hermosa Ave. #7
Hermosa Beach, CA 90254
213-374-3478
Call for your local access number.

Generation Five BBS
(Voice) 301-495-6690
(Data) 301-495-2932
Accessible via PC Pursuit, Washington D.C. From Files submenu, select "P" for Personal Downloads.

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8868

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
Mention *PJ* when you call.

Logical Alternatives Incorporated
Calder Square, P.O. Box 10674
State College, PA 16805-0674
(Voice) 814-234-8088
(Data) 814-234-1881

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

Soft Advances BBS
10811 Washington Blvd. #205
Culver City, CA 90232
(Voice) 213-559-7105
(Data) 213-559-1449

difference: **Alt_TextAttr** is a function and cannot be written to. Instead, use **Alt_Set_TextAttr** and one of its typed attribute specifications. It's easier than figuring out the bit patterns, anyway. **Alt_Get_TextAttr** will return the current video attribute, using the same argument type. Or, if you would rather use "colors," **Alt_TextBackGround** and **Alt_Text_Color** can be applied. *NormVideo* and *HighVideo* also have equivalents, **Alt_NormVideo** and **Alt_HighVideo**, respectively.

The Window procedure in Turbo Pascal is not what most programmers consider a proper window, since it doesn't save the contents of what is underneath it. But it does define a subsection of the physical display to be the current, logical display. **Alt_Window** behaves in the same style, along with **Alt_WindMin** and **Alt_WindMax**, which report the window limits.

DelLine and InsLine are also implemented. **Alt_DelLine** deletes the line the cursor is on and scrolls the lines underneath it to fill the gap. The remaining bottom line will be cleared, using the current video attribute. **Alt_InsLine** performs the reciprocal operation, as it bumps the lines down, scrolling off the bottom line and clearing the current line.

One function provided by AltMon that isn't implemented within Turbo Pascal is **Dual_Monitors_Detected**. This function tests for the presence of two video adapters and returns TRUE if that is the case. This determination is judged on the ability to alter memory locations within the monochrome and color video RAM areas, and it is not foolproof. A Hercules mono card, for example, will pass this test. An alternative might be to use **Swap_Monitor** and then invoke interrupt 10h, service 0Fh, to check and ensure that the mode was really switched.

Listing 2, ALTTEST.PAS *[available on disk or through BBS —Ed.]*, demonstrates most of the features in AltMon. Although it doesn't perform any useful work, it does serve to illustrate how to employ AltMon and gives a good idea of what is possible.

The Integrated Environment

Before concluding, some notes on the Turbo Pascal environment and dual monitors. Invoking the integrated environment with the */d* qualifier will bring up your application on the default monitor and the source code on the alternate display. Enquiring minds want to know "What happens when the default display is switched?" Happily, the answer is "Nothing." The program behaves as intended, and your source code and application program maintain the relationship they inherited when the session started. However, you will not be able to view the application display being built, because the debugger has precedence.

Starting Turbo Pascal in the normal fashion (without the */d* qualifier) will force it to only use one display for the application and source module. However, it will allow you to see both displays, since the application display is always available via the Alt-F5 key.

Conclusion

You are now equipped to exploit the existential pleasures of stereo video. With independently addressable displays, the chains of a solo 80-by-25 monitor loosen perceptibly, and your programming applications enjoy an improved appearance. This is in addition to the bonus of an enhanced program-development environment. Isn't it curious that more machines aren't similarly equipped? PJ

Guy Cole began his programming career seven years ago, while in the U.S. Navy, on a Burroughs mainframe (AN/GYK-3). Since then, he has been employed in DEC PDP-11 and VAX environments and is currently working as a programmer/analyst for Delfin Systems in Sunnyvale, CA. He can be reached on CompuServe at 71046,1405, via BIX as GCOLE, or at P.O. Box 361390, Milpitas, CA 95035.

Code follows.

References

Jourdain, Robert. *Programmer's Problem Solver for the IBM PC, XT & AT*. New York: Brady Communications, 1986.

Wilton, Richard. *Programmer's Guide to PC & PS/2 Video Systems*. Redmond, Wash.: Microsoft Press, 1987.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New application. New network. You know the drill.

You made it. You built a distributed application that takes full advantage of the power of your network. You faced incredible obstacles. And you swore if you survived, you'd never do it again. Guess what?

They want you to do it again.

This time, take a short cut. With Netwise® Our RPC TOOL™ lets you sidestep the obstacles by providing access to a common platform for the development of distributed applications. You write the application, then specify the network. Netwise automatically generates your distributed processing code. All of it. Every time. That means your application can easily migrate to any environment in the future. Banyan, Novell, Sun, 3Com and major software suppliers have endorsed this technology as a standard for building distributed applications. Now it's your turn.

Call us for a free educational diskette. Or order the RPC TOOL for a free thirty day try out. The development tool that lets your application be all it can be.

NETWISE®

Put an end to the grunt work. Call us. 303/442-8280

2477 55th Street, Boulder, Colorado USA 80301

Netwise is a registered trademark. RPC TOOL is a trademark of Netwise, Inc. Banyan is a registered trademark of Banyan Systems Incorporated.
Novell is a registered trademark of Novell, Inc. Sun is a registered trademark of Sun Microsystems, Inc. 3Com is a registered trademark of 3Com Corporation.

Number 278 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from "Cooking with Stereo Video," by Guy Cole

1

```
LISTING 1
Unit AltMon;
{
  Overview/Description:
    AltMon provides support which allows an application to utilize two video
    monitors. Even though it is possible, in some cases, to keep two video
    adapters on the bus, without conflict, MS-DOS and Turbo-Pascal are
    able to utilize only one at a time.

    AltMon gets around this limitation by writing directly to the video RAM
    on the idle monitor. Normal system services on the default monitor are
    not inhibited in any fashion. The net result is two active video
    displays.

    Most of the routines within AltMon are prefixed with an "Alt_", for
    ease in identification and to avoid name conflicts. All of the
    intrinsic Turbo Pascal video-support routines have been implemented.

    For example, to clear the alternate display screen, simply invoke
    "Alt_ClrScr" or to position the alternate cursor, use "Alt_GotoXY(X, Y)".

    Most routines are self-explanatory, but AltMon_Setup must be called
    prior to invoking any of the routines in this module.

  Compiler/Operating System: Turbo-Pascal Ver 5.5 and MS-DOS Ver 3.30

  Maintenance History: 7 June, 1989    G.S.Cole    //Creation//
}
Interface

Uses DOS;

Type
  Alt_String = String[255];

  AltMon_Attribute_Type = (Alt_Normal, Alt_Intense, Alt_Underlined,
    Alt_Intense_Underlined, Alt_Reverse, Alt_Blink, Alt_Blink_Intense, Alt_Blink_Reverse);

{
    Video Attribute Group.
}
Function Alt_TextAttr:Byte;
Function Alt_Get_TextAttr:AltMon_Attribute_Type;
Procedure Alt_Set_Attribute(Candidate:AltMon_Attribute_Type);
Procedure Alt_NormVideo;
Procedure Alt_HighVideo;
Procedure Alt_LowVideo;
Procedure Alt_TextBackGround(Color:Byte);
Procedure Alt_TextColor(Color:Byte);
{
    Cursor Support Group.
}
Procedure Alt_Cursor_Enable;
Procedure Alt_Cursor_Disable;
Procedure Alt_GotoXY(X, Y:Byte);
Function Alt_WhereX:Byte;
Function Alt_WhereY:Byte;
{
    Window Support Group.
}
Function Alt_WindMin:Word;
Function Alt_WindMax:Word;
Procedure Alt_Window(X1, Y1, X2, Y2:Byte);
{
    Display support.
}
Procedure Alt_ClrScr;
```

2

```
Procedure Alt_ClrEOL;
Procedure Alt_DelLine;
Procedure Alt_InsLine;
{
    ...And something to write with...
}
Procedure Alt_Write(Buffer:Alt_String);
Procedure Alt_Writeln(Buffer:Alt_String);
{
    Utilities
}
Function Swap_Monitor:Boolean;
Procedure AltMon_Setup;
{
    Initialization...
}
Function Dual_Monitors_Detected:Boolean;

Implementation

CONST
  Cursor_Attribute = $F0;
  Cursor_Character = $DB;
  Video_RAM = $B000;
  Other_Video_RAM = $B800;

  X_Max = 79;
  X_Min = 0;
  Y_Max = 24;
  Y_Min = 0;

Type
  Display_Record = Record
    C:Char; { the character to be displayed }
    A:Byte; { and it's video attribute }
  end;

  Line_Type = Array[X_Min..X_Max] of Display_Record;
  Display_Type = Array[Y_Min..Y_Max] of Line_Type;

VAR
  Display: Display_Type Absolute Video_RAM:0;
  Attribute_Type:AltMon_Attribute_Type;
  Attribute,
  Cur_X, Cur_Y,
  Cur_X_Min, Cur_Y_Min, Cur_X_Max, Cur_Y_Max:Byte;
  Cursor_Enable_Flag:Boolean;

Procedure Post_Cursor;
{
    Called after moving cursor, ensures (if enabled) a cursor is present.
}
begin { Procedure Post_Cursor }
  If Cursor_Enable_Flag then
    begin
      If Display[Cur_Y, Cur_X].C = ' '
        then Display[Cur_Y, Cur_X].C := Chr(Cursor_Character);
      Display[Cur_Y, Cur_X].A := Cursor_Attribute;
    end;
end; { Procedure Post_Cursor }

Procedure Pre_Cursor;
{
    Invoked prior to moving cursor, ensures it is totally deleted.
}
begin { Procedure Pre_Cursor }
  If Cursor_Enable_Flag then
    begin
      If Display[Cur_Y, Cur_X].C = Chr(Cursor_Character)
        then Display[Cur_Y, Cur_X].C := ' ';
      Display[Cur_Y, Cur_X].A := Attribute;
    end;
end; { Procedure Pre_Cursor }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

```
Procedure Check_4_Scroll;
{
    Determine if vertical scrolling is required, and if so, perform it.
    Scrolling is performed against the logical window boundaries, not
    the physical limits of the display.
}
var
  X, Y:Byte;
begin { Procedure Check_4_Scroll }
  if Cur_Y > Cur_Y_Max then
    begin
      For Y := Cur_Y_Min to Pred(Cur_Y_Max) do        { bump lines up one }
        Move(Display[Y+1, Cur_X_Min], Display[Y, Cur_X_Min],
(Succ(Cur_X_Max-Cur_X_Min)*2));
      For X := Cur_X_Min to Cur_X_Max do              { clear off bottom line }
        begin
          Display[Cur_Y_Max, X].C := ' ';
          Display[Cur_Y_Max, X].A := Attribute;
        end;
      Cur_Y := Cur_Y_Max;
    end;
  Post_Cursor;
end; { Procedure Check_4_Scroll }

Procedure Check_4_CRLF;
{
    Determine if the end of line has been exceeded.
    If so, wrap to the next line, scrolling if required.
}
begin { Procedure Check_4_CRLF }
  If Cur_X > Cur_X_Max then
    begin
      Cur_X := Cur_X_Min;
      Cur_Y := Succ(Cur_Y);
      Check_4_Scroll;
    end;
  Post_Cursor;
end; { Procedure Check_4_CRLF }

Procedure Service_Control_Character(C:Char);
{
    Perform special handling to support the legal control characters.
}
begin { Procedure Service_Control_Character }
  Case C of
    ^G:Write(^G);   { Bell }
    ^H:begin        { Backspace }
         If Cur_X > Cur_X_Min then
           begin
             Pre_Cursor;
             Cur_X := Pred(Cur_X);
             Post_Cursor;
           end;
       end;
    ^J:begin        { Line Feed }
         Pre_Cursor;
         Cur_Y := Succ(Cur_Y);
         Check_4_Scroll;
       end;
    ^M:begin        { Carriage Return }
         Pre_Cursor;
         Cur_X := Cur_X_Min;
         Post_Cursor;
       end;
  end; { case }
end; { Procedure Service_Control_Character }

Function Alt_TextAttr:Byte;
{
    Return the current alternate monitor attribute.
    Note that this differs from the Turbo Pascal implementation in that
    this is a function and cannot be written to (TextAttr is a variable).
    You can manually set video attributes via Alt_TextBackGround and
    Alt_TextColor.
```

4

```
}
begin { Function Alt_TextAttr }
  Alt_TextAttr := Attribute;
end; { Function Alt_TextAttr }

Function Alt_Get_TextAttr:AltMon_Attribute_Type;
{
    Return the current alternate monitor attribute types.

    There is not an equivalent intrinsic Turbo Pascal procedure.
}
begin { Function Alt_Get_TextAttr }
  Alt_Get_TextAttr := Attribute_Type;
end; { Function Alt_Get_TextAttr }

Procedure Alt_Set_Attribute(Candidate:AltMon_Attribute_Type);
{
    Define the current video attribute. Note that these are set with
    typed variables, rather than specifying the bit definitions.

    There is not an equivalent intrinsic Turbo Pascal procedure.
}
begin { Procedure Alt_Set_Typed_Attribute }
  Attribute_Type := Candidate;
  Case Candidate of
    Alt_Normal:     Attribute := 7;
    Alt_Intense:    Attribute := 15;
    Alt_Underlined: Attribute := 1;
    Alt_Intense_Underlined: Attribute := 9;
    Alt_Reverse:    Attribute := 112;
    Alt_Blink:      Attribute := 135;
    Alt_Blink_Intense: Attribute := 143;
    Alt_Blink_Reverse: Attribute := 240;
  end;
end; { Procedure Alt_Set_Attribute }

Procedure Alt_NormVideo;
{
    Set the normal video attributes.
}
begin { Procedure Alt_NormVideo }
  Alt_Set_Attribute(Alt_Normal);
end; { Procedure Alt_NormVideo }

Procedure Alt_HighVideo;
{
    Set highlighted (intense) attributes.
}
begin { Procedure Alt_HighVideo }
  Alt_Set_Attribute(Alt_Intense);
end; { Procedure Alt_HighVideo }

Procedure Alt_LowVideo;
{
    My tests show that this is the same video attribute as NormVideo.
}
begin { Procedure Alt_LowVideo }
  Alt_NormVideo;
end; { Procedure Alt_LowVideo }

Procedure Alt_TextBackGround(Color:Byte);
{
    Insert a new background color into the attribute mask.
}
begin { Procedure Alt_TextBackGround }
  Color := (Color and 7)*16;        { mask background, position for insertion }
  Attribute := Attribute and $8F;   { mask background }
  Attribute := Attribute or Color;
end; { Procedure Alt_TextBackGround }

Procedure Alt_TextColor(Color:Byte);
{
    Insert a new text color into the attribute mask.
    Blinking enabled, if leading bit set.
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Cole code, continued

```
}
begin { Procedure Alt_TextColor }
  Color := Color and $1F;      { mask background, leave blink bit }
  Attribute := Attribute and $70;  { mask foreground, and blink bit }
  Attribute := Attribute or Color; { new result }
end; { Procedure Alt_TextColor }

Procedure Alt_ClrScr;
{
  "Clear" the alternate display by setting the video attributes to normal
  and padding it out with ASCII spaces.
  Exits w/cursor "homed" - respects the logical window definitions.
}
VAR
  X, Y:Byte;
begin { Procedure Alt_ClrScr }
  For X := Cur_X_Min to Cur_X_Max do          { clear off top line }
    begin
      Display[Cur_Y_Min, X].C := ' ';
      Display[Cur_Y_Min, X].A := Attribute;
    end;
  For Y := Cur_Y_Min to Pred(Cur_Y_Max) do      { and copy on down... }
    Move(Display[Y, Cur_X_Min], Display[Y+1, Cur_X_Min], (Succ(Cur_X_Max-Cur_X_Min)*2));
  Cur_X := Cur_X_Min;
  Cur_Y := Cur_Y_Min;
  Post_Cursor;
end; { Procedure Alt_ClrScr }

Procedure Alt_ClrEOL;
{
  Clear the current line, from the current cursor position to end,
  without altering the current cursor location.
}
VAR
  X:Byte;
begin { Procedure Alt_ClrEOL }
  For X := Cur_X to Cur_X_Max do
    begin
      Display[Cur_Y, X].C := ' ';
      Display[Cur_Y, X].A := Attribute;
    end;
  Post_Cursor;
end; { Procedure Alt_ClrEOL }

Procedure Alt_Cursor_Enable;
{
  Enable the pseudocursor on the alternate display.
}
begin { Procedure Alt_Cursor_Enable }
  Cursor_Enable_Flag := True;
  Post_Cursor;
end; { Procedure Alt_Cursor_Enable }

Procedure Alt_Cursor_Disable;
{
  Disable the pseudocursor on the alternate display.
}
begin { Procedure Alt_Cursor_Disable }
  Pre_Cursor;
  Cursor_Enable_Flag := False;
end; { Procedure Alt_Cursor_Disable }

Procedure Alt_GotoXY(X, Y:Byte);
{
  Position the psendocursor. Note that the coordinates are window
  relative, and if invalid, the cursor isn't moved.
}
begin { Procedure Alt_GotoXY }
  X := Pred(X);  { Ref to 0, 0 }
  Y := Pred(Y);
  If (X <= Cur_X_Max) or (Y <= Cur_Y_Max) then
    begin
      Pre_Cursor;
      Cur_X := X + Cur_X_Min;
```

Code continues...

People are talking about us.

"Lahey F77L is definitely for the programmer's market, with features for the casual and professional user...*EDITOR'S CHOICE" PC Magazine*

"It is a fantastic product." *PC Australia*

"This is a very good implementation of FORTRAN, better than the one I use on my minicomputer system." *IEEE Software*

"It should be the last FORTRAN compiler you will need to buy..." *Your Computer*

"Lahey F77L was by far the easiest compiler to use." *Programmer's Journal*

"Compilation speed is Lahey FORTRAN's most remarkable feature; it is unbelievably fast...Realistically, Lahey FORTRAN is the most efficient and productive FORTRAN development tool for the DOS environment..." *Byte*

F77L 4.0 EDITOR, LINKER, PROFILER GRAPHICS, MAKE, WEITEK

When people talk about FORTRAN the name mentioned most often is Lahey Computer Systems, Inc.

Contact us to discuss our products and your needs. (800) 548-4778
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123 Tlx: 9102401256

Number 44 on the Reader Service Card

T-SHIRTS FROM PJ

ISSUE SPECIAL! T-SHIRTS NOW ONLY $10.00

The ***Programmer's Journal*** issue special for Volume 7.6 is **T-Shirts!** Normally $12.00, any of our very fine, high-quality T-shirts can for a limited time be obtained for only **$10.00!** This reduced price still includes postage and handling. **Ordering is easy** with your Visa or Mastercard and our toll free number:

800-234-0386

See our ad on page 96 for details.

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"The only serious technical conference of note for the PC community."

Garry Ray/PC Week

If you can attend only one conference this year, don't miss Computer Language's SOFTWARE DEVELOPMENT '90 — February 6-9, 1990 at the Oakland Convention Center.

A different kind of conference — by and for programmers

At SOFTWARE DEVELOPMENT '90 you'll have three days to focus on practical techniques that will increase your programming power. You'll learn from the most important visionaries and practitioners in software development today. Past speakers include: Larry Constantine, Philippe Kahn, P.J. Plauger, Bill Gates, Bjarne Stroustrup, Ed Yourdon, Steve Mellor, Rich Finkelstein, Robert Ward, Jack Purdum, and dozens more.

Hone your professional skills, investigate new programming tools

Imagine the chance to pick and choose from over 150 lectures and workshops on design methodologies, programming tools and issues, object-oriented programming, C and other programming languages, artificial intelligence, graphics interface design and much more, all tailor-made for you — the experienced software developer.

Share tips and techniques with the "best and the brightest" at organized roundtable discussions, informal evening receptions and vendor hospitality events.

The very latest in development tools and services will be on display at the concurrent Software Development Products Exhibition. Over 100 exhibitors make it the largest trade show exclusively for Software Developers.

Just minutes from downtown San Francisco

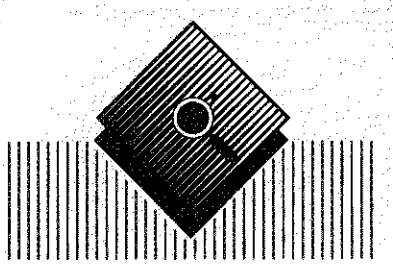
The Oakland Convention Center gives you complete access to the exciting San Francisco Bay Area. From the front door of the center, you're 7 minutes from downtown San Francisco by BART train.

"Probably the most valuable thing I've done in ten years in the software business."

Chuck Atkinson/CAP, Fort Worth, Texas

Register early and save up to 30%

Program details are now being finalized. Return the coupon today for a free information kit. Or call Anastasia Mills at (415) 995-2471. Learn how you can save up to 30% by registering early.



SOFTWARE DEVELOPMENT '90

Oakland Convention Center
February 6-9, 1990

Produced and managed by the publishers of Computer Language, AI Expert, UNIX Review, Database Programming and Design, and Embedded Systems Programming.

Software Development '90
Miller Freeman Publications
500 Howard St., San Francisco, CA 94105
Phone: (415) 995-2471 • FAX: (415) 995-2486

YES!

☐ Please send me a free information kit and registration materials for SOFTWARE DEVELOPMENT '90 to be held February 6-9, 1990 at the Oakland Convention Center.

☐ Please send me information on exhibiting.

PJ 11/12

Name _____

Title _____

Company _____

Street _____

City, State, Zip _____

Phone _____ FAX _____

Software Development '90, Miller Freeman Publications, 500 Howard St., San Francisco, CA 94105

Number 167 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Cole code, continued

6

```
      Cur_Y := Y + Cur_Y_Min;
      Post_Cursor;
    end;
end; { Procedure Alt_GotoXY }

Function Alt_WhereX:Byte;
{
    Return the X coordinate of the cursor.
    Reported position is window relative.
}
begin { Function Alt_WhereX }
  Alt_WhereX := Succ(Cur_X - Cur_X_Min);
end; { Function Alt_WhereX }

Function Alt_WhereY:Byte;
{
    Return the Y coordinate of the cursor.
    Reported position is window relative.
}
begin { Function Alt_WhereY }
  Alt_WhereY := Succ(Cur_Y - Cur_Y_Min);
end; { Function Alt_WhereY }

Function Alt_WindMin:Word;
{
    Return the current upper left corner window coordinates, as packed value.
}
begin { Function Alt_WindMin }
  Alt_WindMin := Cur_Y_Min * 256 + Cur_X_Min;
end; { Function Alt_WindMin }

Function Alt_WindMax:Word;
{
    Return the current lower right corner window coordinates, as packed value.
}
begin { Function Alt_WindMax }
  Alt_WindMax := Cur_Y_Max * 256 + Cur_X_Max;
end; { Function Alt_WindMax }

Procedure Alt_DelLine;
{
    Delete the line at the current cursor location, and scroll up the lines
    underneath it.
}
VAR
  X, Y:Byte;
begin { Procedure Alt_DelLine }
  Pre_Cursor;
  For Y := Cur_Y to Pred(Cur_Y_Max) do
    Move(Display[Y+1, Cur_X_Min], Display[Y, Cur_X_Min], (Succ(Cur_X_Max-Cur_X_Min)*2));
  For X := Cur_X_Min to Cur_X_Max do              { clear off bottom line }
    begin
      Display[Cur_Y_Max, X].C := ' ';
      Display[Cur_Y_Max, X].A := Attribute;
    end;
  Post_Cursor;
end; { Procedure Alt_DelLine }

Procedure Alt_InsLine;
{
    Insert a line at the current cursor location. What this really means
    is bump all the lines down and clear the current line.
}
VAR
  X, Y:Byte;
begin { Procedure Alt_InsLine }
  Pre_Cursor;
  For Y := Pred(Cur_Y_Max) downto Cur_Y do
    Move(Display[Y, Cur_X_Min], Display[Y+1, Cur_X_Min], (Succ(Cur_X_Max-Cur_X_Min)*2));

  For X := Cur_X_Min to Cur_X_Max do              { clear off current line }
    begin
      Display[Cur_Y, X].C := ' ';
      Display[Cur_Y, X].A := Attribute;
```

7

```
    end;

  Cur_X := Cur_X_Min;
  Post_Cursor;
end; { Procedure Alt_InsLine }

Procedure Alt_Window(X1, Y1, X2, Y2:Byte);
{
    Define a portion of the alternate display as the current window.
    All display coordinates will be reference to the new logical coordinates.
    Exits w/cursor "homed".
}
begin { Procedure Alt_Window }
  Pre_Cursor;
  Cur_X_Min := Pred(X1);
  Cur_Y_Min := Pred(Y1);
  Cur_X_Max := Pred(X2);
  Cur_Y_Max := Pred(Y2);
  Cur_X := Cur_X_Min;
  Cur_Y := Cur_Y_Min;
  Post_Cursor;
end; { Procedure Alt_Window }

Procedure Alt_Write(Buffer:Alt_String);
{
    Write a line of data to the display.
}
VAR
  I:Byte;
begin { Procedure Alt_Write }
  Pre_Cursor;
  Check_4_Scroll;
  For I := 1 to Length(Buffer) do
    begin
      If Buffer[I] < ' '
        then Service_Control_Character(Buffer[I])
        else begin
          Display[Cur_Y, Cur_X].C := Buffer[I];
          Display[Cur_Y, Cur_X].A := Attribute;
          Cur_X := Succ(Cur_X);
          Check_4_CRLF;
        end;
    end;
end; { Procedure Alt_Write }

Procedure Alt_Writeln(Buffer:Alt_String);
{
    Write a line of data to the display, followed by a CR/LF sequence.
}
begin { Procedure Alt_Writeln }
  Buffer := Buffer + ^M + ^J;
  Alt_Write(Buffer);
end; { Procedure Alt_Writeln }

Function Dual_Monitors_Detected:Boolean;
{
    Determine if two video adapters are present on the bus. This is
    accomplished by writing to the origin of both the monochrome and
    color video display pages. If I'm allowed to change the contents
    of these memory locations, then two video adapters are probably
    present, and TRUE is returned.
}
VAR
  C:Byte;
begin { Function Dual_Monitors_Detected }
  C := MEM[Video_RAM:0];
  MEM[Video_RAM:0] := Succ(C);
  If C <> MEM[Video_RAM:0] then
    begin
      MEM[Video_RAM:0] := C;
      C := MEM[Other_Video_RAM:0];
      MEM[Other_Video_RAM:0] := Succ(C);
      If C = MEM[Other_Video_RAM:0]
        then Dual_Monitors_Detected := False
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  else begin
    MEM[Other_Video_RAM:0] := C;
    Dual_Monitors_Detected := True;
  end;
end
else Dual_Monitors_Detected := False;
end; {Function Dual_Monitors_Detected }

Function Swap_Monitor:Boolean;
{
  Swap the active display.
  If current display is monochrome, make it color, and return TRUE.
  If current display is color, make it monochrome, and return FALSE.
}
CONST
  Video_Interrupt = $10;
  Equipment_List_Offset = $410;
VAR
  Regs:Registers;
  X:Byte;
begin { Function Swap_Monitor }
  Regs.AH := $0F;     { Determine Video Mode }
  INTR(Video_Interrupt, Regs);
  X := MEM[0:Equipment_List_Offset];
  If Regs.AL = 7 then { Mono Mode? }
    begin  { Set Color Mode }
      X := X AND $CF;
      Regs.AL := 3;
      Swap_Monitor := True;
    end
  else
    begin  { Set Mono Mode }
      X := X OR $30;
      Regs.AL := 7;
      Swap_Monitor := False;
    end;
  MEM[0:Equipment_List_Offset] := X;
  Regs.AH := 0;       { Set Video Mode }
  INTR(Video_Interrupt, Regs);
end; { Function Swap_Monitor }

Procedure AltMon_Setup;
{
  Prepare to use the AltMon module by initializing variables and clearing
    the alternate display.
}
begin { Procedure AltMon_Setup }
  Alt_NormVideo;
  Cur_X_Min := X_Min;
  Cur_Y_Min := Y_Min;
  Cur_X_Max := X_Max;
  Cur_Y_Max := Y_Max;
  Alt_ClrScr;
  Alt_Cursor_Enable;
end; { Procedure AltMon_Setup }

end. { Unit AltMon }
```

LISTING 2

Because of lack of space, Listing 2 is not printed in this issue. It is available on the Code Disk for this issue as well as through our bulletin board services (see pages 54 and 96). —Ed.

U.S. Postal Service

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

Required by 39 U.S.C. 3685)

1A. TITLE OF PUBLICATION	1B. PUBLICATION NO.	2. DATE OF FILING
Programmer's Journal	0 7 4 7 5 8 6 1	09-29-89
3. FREQUENCY OF ISSUE: Bi-Monthly	3A. NO. OF ISSUES PUBLISHED ANNUALLY: 6	3B. ANNUAL SUBSCRIPTION PRICE: $24.00

4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION *(Street, City, County, State and ZIP Code) (Not printer)*: PO Box 30160 Eugene OR 97403

5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER *(Not printer)*: PO Box 30160 Eugene OR 97403

6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR *(This item MUST NOT be blank)*

PUBLISHER *(Name and Complete Mailing Address)*: Elizabeth Oakley PO Box 30160 Eugene OR 97403

EDITOR *(Name and Complete Mailing Address)*: M Steven Baker PO Box 30160 Eugene OR 97403

MANAGING EDITOR *(Name and Complete Mailing Address)*: Robert E Wanetick PO Box 30160 Eugene OR 97403

7. OWNER *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)*

FULL NAME	COMPLETE MAILING ADDRESS
Oakley Publishing Company	PO Box 30160 Eugene OR 97403
Elizabeth Oakley	PO Box 30160 Eugene OR 97403
James Oakley	PO Box 30160 Eugene OR 97403

8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES *(If there are none, so state)*

FULL NAME	COMPLETE MAILING ADDRESS
none	

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES *(Section 422.12 DMM only)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*:

(1) ☐ HAS NOT CHANGED DURING PRECEDING 12 MONTHS (2) ☐ HAS CHANGED DURING PRECEDING 12 MONTHS *(If changed, publisher must submit explanation of change with this statement.)*

10. EXTENT AND NATURE OF CIRCULATION	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES *(Net Press Run)*	21,879	24,326
B. PAID CIRCULATION 1. Sales through dealers and carriers, street vendors and counter sales	6,707	9,280
2. Mail Subscription	10,056	12,380
C. TOTAL PAID CIRCULATION *(Sum of 10B1 and 10B2)*	16,763	21,660
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	949	1,811
E. TOTAL DISTRIBUTION *(Sum of C and D)*	17,712	23,471
F. COPIES NOT DISTRIBUTED 1. Office use, left over, unaccounted, spoiled after printing	3,006	855
2. Return from News Agents	3,161	-0-
G. TOTAL *(Sum of E, F1 and 2 - should equal net press run shown in A)*	21,879	24,326

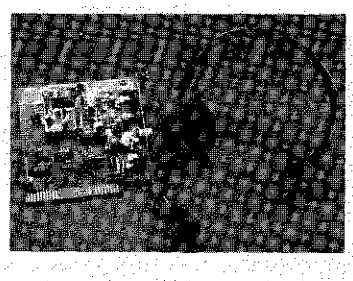
11. I certify that the statements made by me above are correct and complete. SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Elizabeth Oakley Publisher

PS Form 3526 July 1982 *(See instructions on reverse)*

VOICE MASTER KEY®
VOICE RECOGNITION SYSTEM
FOR PC/COMPATIBLES & TANDY 1000 SERIES
A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING. . .The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response. **a complete, superior speech and sound development tool.**



SOFTWARE CONVERSION CODES. . .The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED. . .**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY $149.95 COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS SOFTWARE ONLY--REQUIRES TANDY BRAND ELECTRET MICROPHONE.

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add $5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

COVOX INC.
675 Conger Street, Eugene, OR 97402
TEL. 503-342-1271 • FAX 503-342-1283

Number **277** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Software Engineering Store

C

Borland Turbo C v2 Prof. $175
Blaise C Tools Plus/6.0 ..... $119
Creative Vitamin C $169
Genus PCX Programmer's Toolkit ... $155
read, write, modify & modify ".pcx" graphics files
Gimpel PC Lint $99
Greenleaf Communications Library .. $179
Island Systems C Meta-Menu $129
Magna Carta C Windows Toolkit $89
Microsoft C v5.1 ..... $325
Microsoft Quick C $73
Novell Btrieve $199
Softfocus Btree/ISAM $99
comprehensive file/index manager with all source
Wordcraft C Workshop $69
C training manual & software
Zortech C++ $129
Optimizing C++ Compiler

Other Languages & Tools

Digitalk Smalltalk/286 ..... $164
Borland Assembler & Debugger $105
Borland Turbo Pascal v5.5 Pro $175
Burton Tlib $89
source code management system
Copy Tech Disk Duplicator Pro $129
Genus PCX Toolkit $155
Matrix Layout v2 ..... $175
Microsoft Macro Assembler $111
Microsoft Windows SDK v2.1 ..... $349
Mortice Kern MKS Toolkit v3 ..... $219
KORN shell & 130 UNIX™ -like utilities
Periscope I 512K v4 ..... $599
Polytron Professional PVCS $359
Qualitas 386-Max $69
80386 memory manager
Solution Systems Brief ..... $call
Turbo Power Turbo Professional $99
Whitewater ACTOR $449
High productivity development for MS Windows

Free Catalog & Newsletter

We carry hundreds of products from dozens of publishers. Call or write for your free subscription to our catalog and newsletter.

Returns

We want you to pay for only the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

386 Hardware

The Hauppauge motherboard slips into your XT or AT to provide top 80386 performance with maximum compatibility. The boards work with virtually all 8-bit and 16-bit adapter cards and run standard versions of 16-bit and 32-bit software including DESQview, OS/2, Windows/386 and UNIX.

The motherboards are manufactured in the U.S.A. and are backed by a 30 day money-back guarantee and one year warranty.

386XT w/1MB RAM $1295
20Mhz XT w/2Mb $1875
20Mhz AT w/2Mb $1875
33Mhz AT w/4MB $3595

Hauppauge!

The Software Engineering Store

1-800-443-7176

In Calif: (415)957-0111

Visa • MasterCard
COD • Check

- Most orders shipped within 24 hours
- Free shipping on orders over $100

657 Mission Street, #300-PJ
San Francisco, CA 94105

Programming Classes

The Software Engineering Store offers hands-on training classes in C, Pascal and other languages at our San Francisco store. Please call for a course outline and schedule.

C Windowing

Vitamin C $169
VC Screen $129
top-notch code generator for Vitamin C
DESQview API Ref & C Lib . $179
use internal DESQview functions
Vermont Views $359
complete functionality & great documentation
DataWindows $229
easy-to-use with great documentation & full source
C Windows Toolkit $89
high performance low level functions

Debugging

Periscope v4 *by Periscope*: The Periscope debuggers provide the combinations of hardware and software you need to solve really tough programming problems. Periscope includes all the debugging features you need, but with minimum overhead and maximum flexibility. With Periscope you can debug programs which need all memory, have real-time requirements and/or are picky about vector usage.

Periscope I w/512K $599
0k memory overhead!

Periscope III $1199
hardware breakpoints and trace-back for real-time applications

C Database

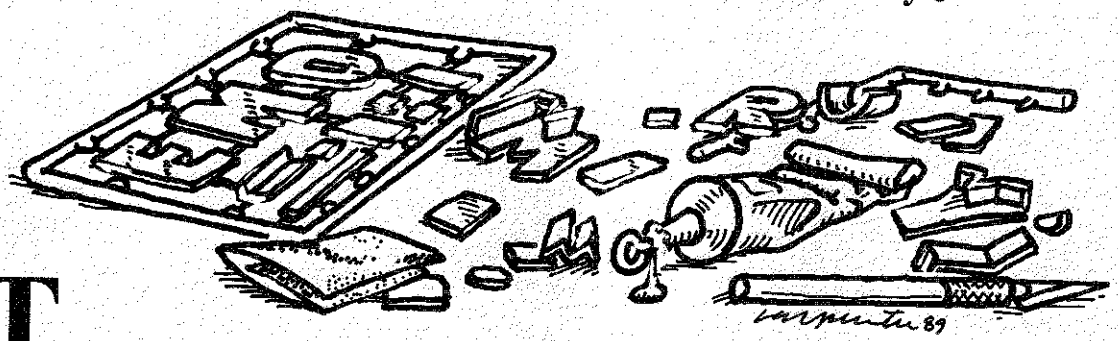
C-Tree $314
highly portable file manager with full source
C-Tree / R-Tree Bundle $519
CQL ..... call
SQL access to C-Tree files
Btrieve $199
Btrieve Network $469
file manager with built-in error recovery & combined data/index files
B-Tree/ISAM $99
highly portable with full source
C-Index for Microsoft C $179
C-Index for Turbo C $89
combined data/index files with great performance and full source
pBase $179
true relational database manager

Prices subject to change without notice • ©1988 TSP. All Rights Reserved • Sep/Oct '89 PJ
Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Efficient 80X86 Memory Models and Static Variables

by John Otken



This column completes my introduction to an efficient methodology for assembly language programs. The first column (*PJ* 7.4, July/August 1989) covered coding conventions; the second (*PJ* 7.5, September/October 1989) covered libraries; and this one covers static and automatic variables, equates, and macros.

The section on static variables introduces techniques that diverge from standard coding practices for high-level languages. At the same time, it offers an excellent solution for one of the most frustrating aspects of assembly language programming. If you are going to restrict the format of your programs to that generated by a C compiler, then perhaps you should code in C. If you code in assembly language because you want smaller and faster programs, you should like this information.

In previous articles, I avoided the topic of code optimization, because good program structure provides the largest performance gain. The real "tricks" to assembly language are those that make programs fastest to write and easiest to maintain. Save the difficult optimizations for the rare sections of code that require it. Without any specific optimizations, your algorithms will be "closer" to the architecture and (it is hoped) more efficient than those generated by a high-level language.

Credit is due to Steve Russell of SLR Systems for many of the fundamental ideas in this column. Steve's programs reflect his tendencies towards optimization. If you have ever used OPTASM or OPTLINK, you know the performance possible from a tightly coded assembly language program.

Static Variables

The two earlier columns developed a memory model similar to Microsoft's small model for assembly language. The two components of this model are the CODE segment and the DGROUP segments. DGROUP consists of segments for initialized and uninitialized static data and the stack.

The "group" aspect of DGROUP does not affect segment ordering. The linker still sequences segments in the normal fashion. References within the segment are affected in that the linker treats DGROUP as one collective segment with no segment boundaries. This combines the advantages of segment ordering within DGROUP and access efficiency of a single segment.

Strict adherence to Microsoft's small model requires that the DS and ES segment registers point to DGROUP. Because my memory model uses a large, dynamic storage system, I cannot guarantee access to static variables in DGROUP via either DS or ES unless you explicitly load them. The direct approach to loading a segment register involves an unappealing three-instruction sequence:

```
B8 9002         mov   ax,@data   ; slow way to
8E D8           mov   ds,ax      ;  access static
8B 0E 0044      mov   cx,count   ;  variable
```

or the smaller but slower sequence:

```
16              push  ss         ; another slow
1F              pop   ds         ;  way to access
8B 0E 0044      mov   cx,count   ;  a static
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Because the stack is part of DGROUP, a more efficient method to access static variables uses a stack-segment override. This not only saves a few bytes but executes faster.

```
8B 0E 0044   mov    cx,ss:count
```

With clever use of the ASSUME directive, the assembler can generate automatic stack overrides. I never use this capability, because too many ASSUME directives ruin code appearance. Each place where DS or ES is modified could require an ASSUME. This includes not only MOV instructions but also any subroutines that return values in DS or ES. Insidious bugs are certain to appear.

```
                assume ds:nothing,ss:dgroup
8B 0E 0044   mov    cx,count   ; automatic override
```

As a general rule, I try to avoid segment overrides in my code. Fortunately, a better method exists for DGROUP access. Indexing off the BP register selects SS as the default segment register. High-level languages access parameters and automatic variables in this way.

Because good assembly language uses registers for parameters and automatic variables, BP is much more valuable as an efficient stack-segment override. Simply point BP to the base of DGROUP, and any instructions that index off BP get "free" access to the stack segment.

The real "tricks" to assembly language are those that make programs fastest to write and easiest to maintain. Save the difficult optimizations for the rare sections of code that require it.

```
8B 8E 0044   mov    cx,count[bp]
```

For this scheme to work properly, DGROUP must align to a paragraph boundary forcing the base offset to zero. Regrettably, the default action is word alignment. To circumvent this, include the following two lines anywhere in the main module of your program:

```
NULL  segment para public 'BEGDATA'
NULL  ends
```

DOS segment ordering (dosseg) forces the linker to position 'BEGDATA' before any other segments in DGROUP. The *para* parameter aligns this segment to a paragraph boundary, forcing all of DGROUP to paragraph alignment. If you forget these lines, DGROUP has an 88-percent chance of paragraph nonalignment, in which case the program will not execute correctly.

Here is a clever technique that takes advantage of short displacements from BP to optimize access to static variables in the first 128 bytes of DGROUP:

```
03 46 44        add   ax,count[bp]
83 C2 00        adc   dx,0
```

When you allocate data in the normal fashion using DB, DW, and DD, the assembler generates long displacements from BP for effective addresses even if the data fall within the first 128 bytes. This happens because the linker—not the assembler— determines data location. However, by moving the data definitions to a structure and allocating the structure as the first data object in DGROUP, the assembler has enough information to generate short displacements.

This technique works with multi-module programs, too. Place the structure definition in an include file, and your program can access the members from any module. Be sure to allocate space for the structure in the first module only; otherwise, your program will not work.

The beauty of this optimization is that you do not have to apply it until you finish the program. At the point when a program is more or less working, define a **dgroup_str** structure, move important static definitions to it, allocate the structure as the first DGROUP object, and fix extern directives as necessary. Voila, the program is smaller and faster!

The program in Listing 1 demonstrates use of a DGROUP structure. Because this program is short, all static variables fit in the structure. Longer programs can allocate data outside the structure using DB, DW, and DD directives as long as **dgroup_str** appears at offset zero of DGROUP. The sample program also initializes statics in the structure definition, which works fine, although I am somewhat uncomfortable with the idea.

When you use a DGROUP structure, you must adhere to the following rules. The first two apply to DGROUP data in general.

- Remember to zero BP during start-up and explicitly set the stack pointer to compensate for MS-LINK bug.
- Do not forget to append "[bp]" to DGROUP references.
- Use the DOSSEG directive to guarantee segment order in DGROUP.
- Do not use constant (.const) and initialized data (.data) in programs that use a DGROUP structure. The linker does not guarantee the order of these segments, and space for the DGROUP structure must be allocated at DGROUP offset 0.
- You must allocate space for the DGROUP structure as the first object of the data segment in the first module only!
- Use only one DGROUP structure per program. If you use different structures with each module, your program is guaranteed to fail. A single DGROUP structure definition can be shared through an include file.
- Write your program first and then move commonly accessed data to the DGROUP structure.
- You must include the 'BEGDATA' segment directives to force paragraph alignment of DGROUP.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

epsilon
programmer's editor

State of the Art editor for DOS, OS/2, Xenix & Unix for PC/XT/AT's

Version 4.0 New!

1988 BYTE AWARD OF DISTINCTION

". . . these products–every one of them–are ones we wouldn't hesitate for a moment to recommend." (Byte, Jan. 1989)

Unrivaled Undo AND Redo

Epsilon gives you the most complete Undo and Redo of any PC programmer's editor. You can undo any sequence of changes with a touch of the Undo key. In fact, you can undo the last few hundred or so operations–you set the limit. And unlike the "full" undo offered by some other editors, Epsilon has Redo, so you can undo your undo's. You can even redo your undo's, and undo your redo's! This flexibility keeps you from losing your work by undoing too much.

Ultimate Customizability . . .

Epsilon features the most powerful extension language of any PC editor. We call it EEL, for Epsilon Extension Language. EEL gives you the rich syntax and data types of C, so you can easily extend or customize the editor to suite your taste. EEL is a real language, not an afterthought. In fact, all of Epsilon's commands are written in EEL, and you get complete source code to all the commands!

". . .but the pièce de résistance is EEL, a C-like extension language that comes with its own compiler. Lugaru Software has implemented Epsilon's entire user-interface in EEL so you can, for example, create your own syntax-sensitive modes or program the help system to browse through your source libraries." (Byte, Jan. 1989)

- C-like extension language, called EEL
- Source code to all commands
- Convenient multilevel undo plus total redo
- Scans compiler errors (customizable)
- Swaps out of memory for big compilers
- Unlimited file size, number of files
- On-line tutorial for quick startup
- As many windows as will fit on the screen
- Supports large displays (like VGA & EGA)
- Context sensitive help autoconfigures
- Tags package indexes subroutine definitions
- Files can be larger than available memory
- Fully utilizes EMS memory
- Interactively customizable keyboard
- EMACS-style command set
- Record keystroke sequences
- Multitasks compilers, even under DOS
- Language support (C, Lisp, etc.)
- Completion on file names, commands, tags, ...
- Full command line editing
- Optional backups
- Regular expression search and tagged replace
- Source level debugger & profiler for EEL
- Do interrupts from EEL (DOS)
- Call any dynamic-link library for EEL (OS/2)
- Saves deleted text (n times)
- Navigate your directory structure
- Not copy protected, for your convenience

. . . & True Multitasking Compiles!

Epsilon's exclusive multitasking facility lets you run your compilers, utilities, or even the DOS command processor, in an Epsilon window, WHILE YOU EDIT. You'll be amazed to see your compiler chugging away while you continue to edit your files. No more dead compile time! If any compiler errors appear, just touch a key and Epsilon parses the error message, then jumps right to the offending line–while your compiler is still running! Don't be fooled by misleading claims from other vendors: Epsilon is the ONLY editor for DOS with this feature! Of course, Epsilon's multitasking also works under OS/2, Xenix, and Unix.

"This feature is especially useful in a programming environment because compilers also can be run in the process buffer. . . If the screen is split into multiple windows, one of which is the process window, the user can see the compilation output while continuing to edit... A single command takes the user to the next error line in a source file and displays the error on the bottom of the screen." (PC Tech Journal, May 1987)

Ask about our free trial!

$195 direct — 60-day money-back guarantee!

Order Hotline: (412) 421-5911

FAX: (412) 421-6371

Lugaru Software Ltd. / 5843 Forbes Avenue / Pittsburgh, PA 15217

Number 271 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Automatic Variables

Every assembly language programmer encounters the dreaded situation when an algorithm just does not work with the number of available registers. Three techniques are useful in such a situation.

The first technique is to try using general-purpose static variables. My programs usually define "t1", "t2", "t3", and "t4" as general-purpose register extensions. The locations are defined within the first 128 bytes of BP and offer the next-best thing to register efficiency. If the routine needs to be reentrant and more than one or two locations are required, you should consider other methods.

The second technique is good, old-fashioned, high-level-language stack variables. This method saves BP, moves the current stack pointer to BP, and subtracts from the stack pointer to create an area of free storage. To access the storage, apply a negative index to BP. More-advanced processors such as the 286 have a single-instruction ENTER that performs these steps.

This method requires the routine to be mostly self-contained because access to DGROUP via BP is unavailable (as are any library routines that access DGROUP in this way). The advantage of this method lies in standardization: Many programmers understand this technique.

The tricky part of the standard technique is defining local variables. MASM 5.1 has a nifty LOCAL directive—but it does not work, which practically ruins its use for any programs that might accidentally be compiled with MASM 5.1.

Structure definitions almost work, but the offsets must be negative, not positive. You might think you could trick the assembler by using a structure of the following form:

```
foo_autos struc
        db      -4 dup(?)
  i     dw      ?
  j     dw      ?
foo_autos ends
```

This works with OPTASM but not with TASM or MASM. About the only portable way to accomplish this is with individual equates:

```
i     equ     -2
j     equ     -4
;
foo   proc
      push    bp
      mov     bp,sp
      sub     sp,4
      mov     i[bp],ax
      mov     j[bp],dx
      ...
      mov     sp,bp
      pop     bp
      ret
foo   endp
```

The third technique for automatic variables (which I favor over the second) is to allocate and access storage in a stack frame without modifying BP. The trick is to double-index BP with another register, either DI or SI. A significant advantage to this method is that variable allocation uses a simple structure definition.

```
foo_autos struc
  i  dw  ?
  j  dw  ?
foo_autos endp
;
foo     proc
        sub     sp,size foo_autos
        mov     di,sp
        sub     di,bp
        mov     i[bp+di],ax
        mov     j[bp+di],dx
        ...
        lea     sp,[bp+di+size foo_autos]
        ret
foo     endp
```

You lose another register (usually DI) with this method, but new variables are trivial to create in the structure definition. Study the code and try a few routines under the control of a debugger.

All three techniques are useful in certain circumstances. In choosing between the second and third, I prefer the third, because structure definitions are convenient and the method has no restrictions on the library interface.

Signed and Unsigned Conditionals

As a warm-down from the above brain "workout," I offer the short and sweet topic of signed versus unsigned conditionals. When you program a magnitude comparison in C, the compiler decides what type of conditional jump to use on the basis of the declared types of the operands.

Except in the case of a mixed comparison of signed and unsigned values, the compiler's choice should be correct. In assembly language, the programmer must make all decisions on conditional jumps. In my experience, unsigned conditionals are the vast majority. In either case, the 80X86 offers a full complement of conditional branches for signed and unsigned comparisons:

unsigned	signed	meaning
JA *target*	JG *target*	greater than
JAE *target*	JGE *target*	greater than or equal to
JBE *target*	JLE *target*	less than or equal to
JB *target*	JL *target*	less than

The mental tip for reading compare and jump instructions is to replace the "cmp" with the word *if*, replace the comma between the compare's operands with the equivalent phrase, and then "goto" the target of the jump. For example:

```
cmp     ax,dx
jb      fool    ; if ax < dx goto fool

cmp     si,dx
jae     foo4    ; if si >= dx goto foo4
```

These are not examples of good commenting practice (see my column in *PJ* 7.4).

Macros

A feature available in more-advanced assemblers, macros generate one or more lines of arbitrary assembly language from a single source line. With the power to

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Commenting Disassembler!

SOURCER™ 486

- SEE HOW PROGRAMS WORK
- EASILY MODIFY PROGRAMS

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80386/387, 80486, and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

Sourcer is the best disassembler we've ever seen!
—PC Magazine, January 17, 1989, page 101

SAMPLE OUTPUT

Fully automatic

Program header

Assembler directives

Determines data areas and type

Detailed comments

Simulator follows segment changes

80386 and 80486 support

Easy to read format

```
resetprn.lst    ResetPRN v1.02                Sourcer Listing    18-Sep-89  1:42 pm      Page 1

                                PAGE  60,132
                                ;-----------------------------------------------------------------------------
                                ;                               RESETPRN
                                ;
                                ;       Created:   24-Aug-89
                                ;       Version:   1.02
                                ;       Passes:    8          Analysis Flags on: H
                                ;-----------------------------------------------------------------------------

                                .386c

= 0008                          @prn_port_1     equ     8                       ; (0040:0008=378h)

                                ;-------------------------------------------------------------------- seg_a ---
                                seg_a           segment para use16 public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                resetprn        proc    far
658E:0000                       start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.02', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:0010  30 32 0D
658E:0013  0040                 data_2          dw      40h
658E:0015  0D 0A 52 65 73 65    data_3          db      0Dh, 0Ah, 'Reset Printer? $'
658E:001B  74 20 50 72 69 6E
658E:0021  74 65 72 3F 20 24
658E:0027                       loc_1:
658E:0027  0E                                   push    cs
658E:0028  1F                                   pop     ds
658E:0029  BA 0015                              mov     dx,offset data_3        ; (658E:0015=0Dh)
658E:002C  B4 09                                mov     ah,9
658E:002E  CD 21                                int     21h                     ; DOS Services  ah=function 09h
                                                                                ;  display char string at ds:dx
658E:0030  B4 01                                mov     ah,1
658E:0032  CD 21                                int     21h                     ; DOS Services  ah=function 01h
                                                                                ;  get keybd char al, with echo
658E:0034  3C 79                                cmp     al,79h                  ; 'y'
658E:0036  75 1A                                jne     short loc_3             ; Jump if not equal
658E:0038  8E 1E 0013                           mov     ds,data_2               ; (658E:0015=40h)
658E:003C  8B 16 0008                           mov     dx,ds:@prn_port_1       ; (0040:0008=378h)
658E:0040  83 C2 02                             add     dx,2
658E:0043  B0 08                                mov     al,8
658E:0045  EE                                   out     dx,al                   ; port 37Ah, printer-2 control
                                                                                ;  al = 8, initialize printer
658E:0046  66: B9 00020000                      mov     ecx,20000h
658E:004C                       loc_loop_2:
658E:004C  67: E2 FD                            loopd   loc_loop_2              ; Loop if ecx > 0
658E:004E  B0 0C                                mov     al,0Ch
658E:0050  EE                                   out     dx,al                   ; port 37Ah, printer-2 control
                                                                                ;  al = 0Ch, init & strobe off
658E:0051                       loc_3:
658E:0051  B4 4C                                mov     ah,4Ch                  ; 'L'
658E:0053  CD 21                                int     21h                     ; DOS Services  ah=function 4Ch
                                                                                ;  terminate with al=return code
                                resetprn        endp
                                seg_a           ends

                                ;-------------------------------------------------------------- stack_seg_b ---
                                stack_seg_b     segment para use16 stack
6593:0000  00C0[FF]                             db      192 dup (0FFh)
                                stack_seg_b     ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

- CHANGE AND ADD FEATURES
- CLARIFY INTERFACES

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER Commenting disassembler	$99.95	**UNPACKER**™ Unpack packed EXE files and more	$39.95
SOURCER with BIOS Pre-Processor	139.95	**ASMtool**™ Assembly source analyzer and flowcharter	89.95

Shipping & Handling: USA $3; Canada/Mexico $10; Other $15; CA Res. add sales tax; PS/2 trademark of IBM Corp

NOT COPY PROTECTED **30-DAY MONEY-BACK GUARANTEE**

If within 30 days of purchase you find our product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

For orders and information, call:

VISA **1-800-662-8266** MasterCard

V COMMUNICATIONS, INC.

3031 Tisch Way, Suite 802, Dept. PJ4, San Jose, CA 95128 (408) 296-4224

Number **125** on the Reader Service Card

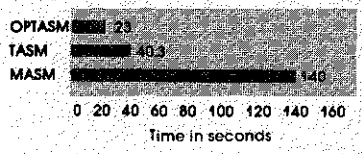
THE PROGRAMMER'S CHOICE

OPTASM

- fastest assembler available
- eliminates phase errors
- never inserts spurious NOPs
- complete TSR online help
- includes a linker and debugger
- "OPTASM is clearly technically superior to MASM"
  Computer Language - Dec 88

$125

ASSEMBLING BGIDEMO ON PC

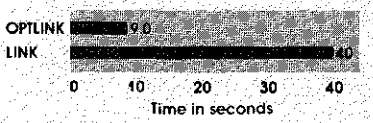


OPTLINK

- fastest linker available
- generates EXE, COM or SYS
- generates global XREF
- supports CodeView info
- works with MS C5.1

$125

LINKING BRIEF

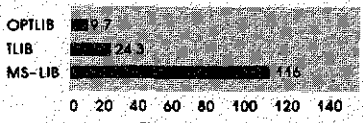


OPTLIB

- 10 times faster than LIB
- * and ? in filenames
- time-based replace
- LIB-compatible syntax
- rename PUBLICs and EXTRNs

$49

BUILD 670K .LIB FILE



OPTDEBUG

- source level - supports CV info
- online help system
- reverse execution
- 50k COM file
- debug EXEs, COMs and TSRs

$125

SLR SYSTEMS

1622 N. Main Street Butler, PA 16001

412-282-0864 TELEX 559215

800-833-3061 (USA)

Call *today* to order!

Number **165** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

manipulate parameter lists, local labels, and conditional assembly, macros offer a simple mechanism to create complex sequences of code. Macros also provide an easy way to obscure algorithms and render a program useless and unmaintainable.

If you need convincing, check out the cmacros.inc file in the Microsoft C start-up code. This file has no documentation and is full of such stuff as the following fragment:

```
?pp macro nL,t,l,s
if ?CPD
.xcref
ife ?PLM
irp x,<nL>
?pp1 x,<t>,%?PO,%?adj,%(?PO+?adj)
?PO = ?PO + 1
.xcref ?T&&x
?T&&x = s
endm
```

Talk about job security. No decent programmer would ever work on stuff like this!

Because macros are never documented as thoroughly as normal instructions, their exact functions can be a mystery—even to the author. While you can expect a programmer to know what an instruction does, you cannot expect him or her to know or understand your macros.

Macros do not save programming time, because typing out the equivalent macro expansion does not take very long, and the few seconds of typing you save are lost many times over when you have to look up a macro definition because you can't recall how it works.

In spite of their dark side, macros have a useful purpose. They can increase code readability and maintainability. The secret is to keep your macros simple and self-documenting. For example, I have a MOVX macro that permits me to code

this	instead of this
movx ax,0	xor ax,ax
movx ds,es	push es
	pop ds

Another macro implements a push-multiple-registers "instruction." Use this macro to avoid multiple lines of push instructions at the beginning of routines:

```
pushm   macro   a,b,c,d,e,f,g,h
        irp     x,<a,b,c,d,e,f,g,h>
        ifnb    <x>
        push    x
        else
        exitm
        endif
        endm
        endm
```

These macros increase code clarity through simplicity.

Here are some general rules for macros:

- Use macros to simplify code and increase readability.
- A macro's function should be obvious and self-documenting.
- Do not use macros as code "shorthand."
- Limit the number of macros.
- Avoid distributing macros in include files.

Conditional Assembly

As with macros, avoid conditional-assembly directives whenever possible. When you use conditional assembly, you lose correspondence between source and object files. Consider the following module, foo.asm:

```
        title   foo
        .model small
        .code
        public foo
foo     proc
        ifdef   flg
        mov     ax,-1
        else
        mov     ax,0
        endif
        ret
foo     endp
        end
```

Because **foo**'s behavior depends on conditional assembly, you cannot be certain whether **foo** returns 0 or -1; you cannot (easily) tell from the OBJ file. Mistakes are bound to result: Even if you explicitly reassemble foo.asm, you might misspell "flg."

Sometimes conditional assembly is useful; for example, I have start-up code for embedded systems. I prefer to use conditional assembly rather than maintain separate versions for each memory model. Because the program displays a message indicating the selected memory model during assembly, problems with misspelling are unlikely.

Equates

The purpose of equates is to increase both the readability and the maintainability of programs. Instead of sprinkling random constants throughout a program, equates provide mnemonic meaning to numbers.

Not that every number stands to benefit from an equate. Good candidates for equates include function codes for DOS and BIOS interrupts and important program constants such as buffer size. The following are some equates for interrupt vectors from the *IBM Technical Reference:*

```
INT_VIDEO_IO        equ     10h
INT_MEMORY_SIZE     equ     12h
INT_KEYBOARD_IO     equ     16h
INT_PRINTER_IO      equ     17h
```

I also like equates for "special" numbers such as TWENTY_FIVE, FORTY_THREE, and FIFTY. These equates are excellent search targets when I need to find code that works with screen dimensions.

```
TWENTY_FIVE     equ     25      ; these equates are
FORTY_THREE     equ     43      ; search targets for
FIFTY           equ     50      ; screen code
```

Bad candidates for equates include constants used only once within a program. For example, a STACK_SIZE equate used once as the argument to a **stack** directive is a poor equate, because a programmer must needlessly look in two places for a single piece of information.

Another bad place for equates is "common knowledge" sections of short programs. Any benefit

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

equates would provide to the following fragment are more easily provided with simple comments:

```
mov     ah,9            ; display hello_text
lea     dx,hello_text
int     21h
```

I do not believe a novice programmer would find the following code easier to understand:

```
mov     ah,DOS_DISPLAY_STRING
lea     dx,hello_text
int     MS_DOS
```

A large program might benefit from these equates, but a short program only becomes a long one when "over-equated."

The C language convention of uppercase equate names is good because it makes equates identifiable in programs. Equates belonging to the same group should begin with a unique prefix; for example, DOS function codes might start with "DOS_". A truly unique prefix has the advantage that a text search for the prefix does not find many extraneous references. For that case, "DOS_" is a poor choice, because those characters are likely to appear in many symbols.

General-purpose text equates are a handy technique for program simplification. My internal programs define several extended registers that are actually static variables. I prefer to access these locations as "t1" and "t2" instead of "t1[bp]" and "t2[bp]". So, I define the locations as "t1_bp" and "t2_bp" and then define text equates:

```
t1  equ  t1_bp[bp]
t2  equ  t2_bp[bp]
```

Now the assembler substitutes the full symbol expression.

Another use for text equates is to mark certain instructions. For example, I have simple equates for "lds" and "les" mnemonics that mark instructions I know are going to break under OS/2:

```
lds_  equ  lds  ; use lds_ and les_ mnemonics
les_  equ  les  ; to mark broken OS/2 code
```

Right now, the equates have no effect on the program, but they will facilitate the transition to OS/2 when the time comes.

In theory, the assembler should replace the symbol with the equated text wherever the symbol appears. Not true. In what can only be described as a misguided quest for speed, MASM neglects to search the symbol table in certain places.

Using the aforementioned start-up code as an example, I want to define a text equate "near_far" to either "near" or "far," based on the current memory model. This way, I could define procedures as follows:

```
foo  proc  near_far
     ...
     ret
foo  endp
```

OPTASM and TASM handle this case, but MASM fails. I can resort to a macro definition to solve the problem

ASMFLOW $99.95

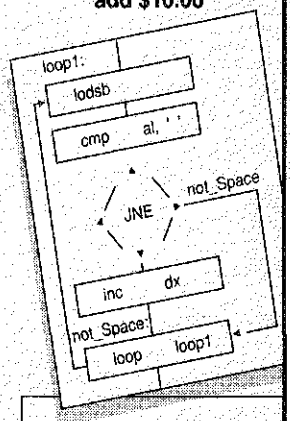
AT LAST!

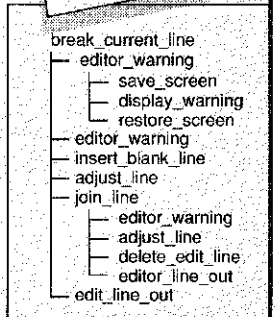
S/H $3.00
Outside U.S.
add $10.00

YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE

FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE

- Flow Charts
- Tree Diagrams
- Stack Sizing
- Register Analysis
- CPU Timing Analysis
- Procedural X-Reference
- 8088/87 to 80386/387
- Context-Sensitive Help
- Menu/Batch/Command line Operation
- MASM 5.1 Compatible





QUANTASM CORPORATION
19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826

VISA · MC 30 - Day Money Back Guarantee

Number 252 on the Reader Service Card

The Indispensible Quick Reference Guide for Assembly Language Programming

MASM QuickRef

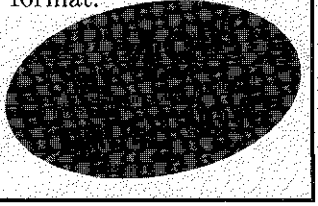
- MASM Instruction Set
- Video Interrupts
- Keyboard Scan Codes
- Display Attributes
- Shift Mask Codes
- Hex-Dec Binary Conversion

FREE Demo Disk Available

Now Also Includes Programmer's Calculator

Memory Resident — Pop-Up Windows

"It's a friendly tool that lets you quickly find commands and procedures in an organized, concise pop-up window format."



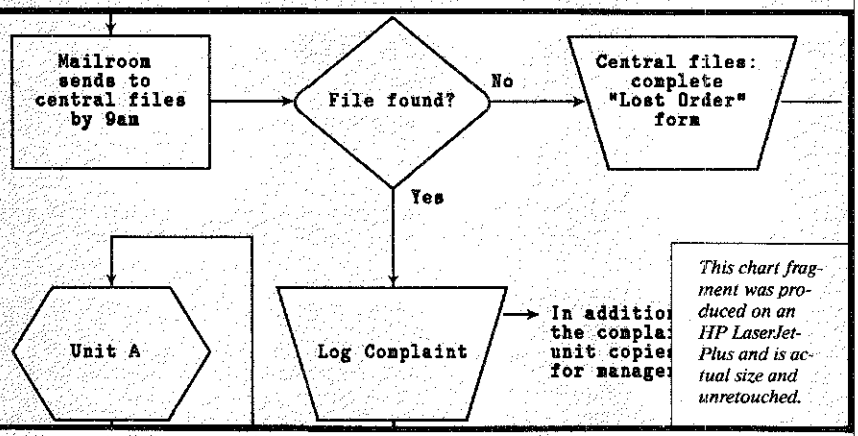
Ultra-X, Inc.
2005 De La Cruz Blvd.
Suite 115
Santa Clara, CA 95050
800-722-3789
In CA: 408-988-4721
Fax: 408-988-4849

Number 290 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EasyFlow

An on-screen chart processor that knows about charts and diagrams, not just another "screen-draw" program that makes you do most of the work. **EasyFlow** is a powerful full-screen graphics program dedicated to data-flow diagrams, flow charts and organization charts. With it you can quickly compose diagrams. More important, you can easily modify them so they are always up to date.



Automatic. Fully automatic text centering within shapes. Fully automatic line routing & re-routing. Fully automatic placement of text along lines.

Fast. Written in assembly language for speed.

Large. Diagram size limited only by available memory. Supports LIM expanded memory. Diagram too large for your printer? EasyFlow automatically breaks the diagram up & prints it in page size pieces.

Standard. All standard data flow and flowcharting shapes included. Other shape libraries are available.

User friendly. Don't take our word for it. PC Magazine says *"EASYFLOW lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts"*.

Mouse. Optional but fully supported.

It works. Over 50,000 copies sold. Two of the "big eight" accounting firms have world wide site licenses for the use of EasyFlow.

It prints. On most popular printers including IBM, Epson, Toshiba, HP LaserJets, HP Desk Jet XEROX 4045, postscript printers and many others.

It plots. On HP7440, 7475, 7550 and 7585B plotters.

Documented: 130 page manual plus over 150 screens of context sensitive help.

Rush delivery: Order by noon today (eastern time) and you'll have it tomorrow. Rush delivery charge is $10.50 (instead of $2.00) and is available only to continental USA and Canada.

EasyFlow works on IBM PC's, IBM PS/2 and compatibles. Requires 512K memory, hard disk and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., check, VISA MasterCard or Company PO.

Haven Tree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

HAVENTREE SOFTWARE LIMITED

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext. 31
Telefax(G3): (613) 544-9632

Number **154** on the Reader Service Card

and obscure the code at the same time; instead, I put nasty comments in the code.

The important points to remember about equates are:

- Equates fix the value of a constant to a single program line, which simplifies program maintenance.
- Equates help document code and increase readability.
- Equates can group or connect similar constants, usually with a unique prefix.
- Equates provide good text-editor search targets.
- Numeric equates should use uppercase symbols.

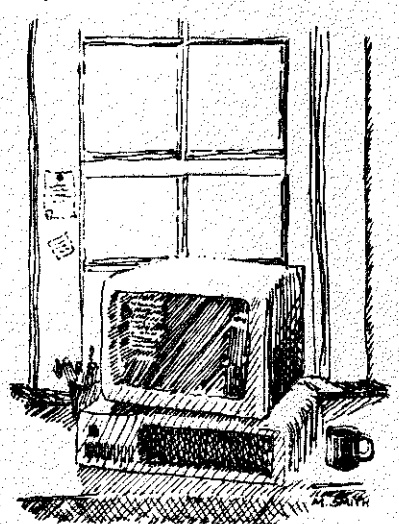
Final Touches

Keep those calls coming into the bulletin board system—the number is 213-559-1449. Files related to this column are pj74.zip, pj75.zip, and pj76.zip.

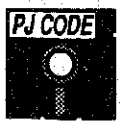
I may be losing the lexical war for proper usage of *assembly* versus *assembler*. In the last two months, I noticed a column in *Computer Shopper*, a Microsoft ad, and the title of a soon-to-be-published book that misuse the word *assembler*. You would sound silly if you said you program "in C compiler"; so why insist on programming "in assembler"? I choose this as my crusade because, as in crusades in times past, the cause is noble and the chance of success nil. PJ

When he isn't hacking assembly language or writing this column, John Otken trains for triathlons. At the USTS race in San Diego, he swam 1 mile, biked 24 miles, and ran 6 miles in 2 hours and 18 minutes (a personal record). John may be reached at 10811 Washington Blvd., Suite 205, Culver City, CA 90232, 213-559-7015.

Code follows.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from ". . . 80X86 Memory Models . . . ," by John Otken

1

LISTING 1

```
          dosseg
          .model    small
          .stack
NULL      segment para public 'BEGDATA'     ; force paragraph
NULL      ends                              ; alignment in DGROUP

        .data
dgroup_str struc
  ran_num       dw        0         ; random number seed
  ran_mod       dw        0FFEFh    ;  modulus      2^16-17
  ran_mul       dw        0FFD9h    ;  multiplier   2^16-39
  ran_inc       dw        0FFFFh    ;  increment    -1
dgroup_str ends

dgroup_start    dgroup_str <>       ; allocate space for dgroup structure

        .code
        extrn   exit_program:near,outchr_newline:near,outchr_unsigned:near

;;      main
;
main    proc
        mov     ax,@data                ; compensate for MS-LINK
        mov     ss,ax                   ;  bug which fails to set
        mov     sp,offset stack         ;  DGROUP relative SS:SP

        cld                             ; "start up" code for this
        mov     bp,offset dgroup_start  ;  memory model

        mov     cx,10                   ; display some random numbers
mai1:   call    random_number
        call    outchr_unsigned
        call    outchr_newline
        loop    mai1

        jmp     exit_program
main    endp


;;      random number
;
;       exit    AX      random number
;
random_number proc
        push    dx                      ; compute random number using
        mov     ax,ran_num[bp]          ;  linear congruential method.
        mul     ran_mul[bp]             ;  see Knuth: Seminumerical Algorithms
        add     ax,ran_inc[bp]
        adc     dx,0
        div     ran_mod[bp]
        xchg    ax,dx
        mov     ran_num[bp],ax
        pop     dx
        ret
random_number endp

        end     main
```

LISTING 2

```
;
; put these routines into individual library modules
;
        .model  small
        .code
        public  exit_program,ms_dos,outchr,outchr_newline,outchr_unsigned

;;      exit program
;
exit_program proc
        mov     ax,4C00h
        jmp     ms_dos
exit_program endp


;;      ms dos
;
ms_dos  proc
        int     21h
        ret
ms_dos  endp


;;      outchr
;
;       entry   AL      character
;       uses    AX
;
outchr  proc
        push    dx
        mov     ah,2
        mov     dl,al
        call    ms_dos
        pop     dx
        ret
outchr  endp


;;      outchr newline
;
;       uses    AX
;
outchr_newline proc
        mov     al,'M'-'@'
        call    outchr
        mov     al,'J'-'@'
        jmp     outchr
outchr_newline endp


;;      outchr unsigned
;
;       entry   AX      number
;       uses    AX
;
outchr_unsigned proc
        push    cx
        push    dx

        mov     cx,10
        mov     dx,-1                   ; push sentinal
ocu1:   push    dx
        xor     dx,dx
        div     cx                      ; divide & push decimal digits
        and     ax,ax
        jnz     ocu1
        inc     dx
ocu2:   xchg    ax,dx                   ; pop & display digits
        add     al,'0'-1
        call    outchr
        pop     dx
        inc     dx
        jnz     ocu2                    ;  if not sentinal
        pop     dx
        pop     cx
        ret
outchr_unsigned endp

        end
```

2

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
NEW!	CSource Application Program Generator by MBAC (includes *all* source code; generator & libraries)	$500
	dB2c (dBase-to-C translator; includes dB_Files for C and dB_Tools for C)	$325
	pBase (relational DBMS with debugging calls and sparse table & repeated field support)	$325
	CQL Query System (SQL retrievals on B-trees plus windows)	$325
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
NEW!	Drasch Clisp with Crules (Lisp library and programming environment with rule processing capability; natural language example)	$300
	Greenleaf Data Windows (windows, menus, data entry, interactive form design; specify compiler)	$300
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$290
	Code Base (database manager, dBase and Clipper compatibile indes & data files; Version 4.0)	$260
	MEWEL (extensible window and event library by Magma Software; message-passing & object-oriented; SAA-compatible; dialog editor)	$250
	TurboTEX (Release 2.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
NEW!	PC PostScript (*complete* PostScript interpreter (ROM Version 47.0A), 80286/386 only, many device drivers, optimized graphics, fast)	$250
	db_File & db_Retrieve (B-tree and network database with SQL query and report writer)	$245
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	$225
NEW!	CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	$210
NEW!	BCPL Compiler (this is *not* C source but BCPL source; BCPL is the mother of C)	$195
	QuickGeometry Library (large collection of mathematics, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC)	$170
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$165
	TurboGeometry (library of routines for computational geometry, Version 3.0)	$160
	AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
NEW!	Cephes Mathematical Library (over 100 high-quality, double-precision scientific functions)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
NEW!	vLIB (270 C functions for windows, menus, forms, pop ups, mouse support, and input editing)	$125
NEW!	Power Search by Blaise Computing (regular-expression compiler; generates machine code on the fly)	$120
	Install 2.3 (automatic installation program; user-selected partial installation; CRC checking)	$120
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$105
Updated!	Minix Operating System (Version 1.3; U**x-like operating system, includes manual)	$105
	HyperText Viewer (simple hypertext system; multi-file documents; includes Tiny Curses)	$100
	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, update by Dan Lanciani)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Otter 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	Make (macros, all languages, built-in rules)	$75
	eval0 (C function to evaluate ASCII infix expression string; 17 built-in functions)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	ScreenLib (simple screen definitions, windows, pop-up menus, context-sensitive help)	$65
	Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	$65
	SYSKIT (rommable or TSR debug/monitor; easily expanded)	$60
	Quincy (interactive C interpreter)	$60
	Symtab (general-purpose symbol table construction and management package)	$60
	PTree (general-purpose parse tree construction and management package)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	$55
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
NEW!	REGX Plus (search and replace string manipulation routines based on regular expressions)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; does not include modem driver)	$50
Updated!	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	$50
	Fortran-to-C Translator by Polyglot	$40
NEW!	FlexList (doubly-linked lists of arbitrary data with multiple access methods)	$40
	Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
	OOPS (collection of handy C++ classes by Keith Gorlen of NIH; Version 2.2)	$35
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor; now includes documentation)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
NEW!	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	$35
	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
NEW!	GNU Awk & Diff for PC (both programs in one package)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	$25
Updated!	FLEX (fast lexical analyzer generator; new, improved LEX; official BSD Version 2.1 with docs)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
	Data	
	Protein Sequences (over 10,000 sequences; includes demo disk of Pearson FAST/A programs)	$60
	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	$60
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	$55
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
	USNO Interactive Computer Ephemeris (high-precision moon, sun, planet & star positions)	$30
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

Number 30 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Aztec C and the ANSI C Standard Delayed

by Rex Jaeschke

© 1989 by Rex Jaeschke. All rights reserved.

In this installment, I report on the status of the draft ANSI C standard. I also look at the Aztec C compiler, from Manx.

Status of ANSI C

(The following is taken from a mailing by Jim Brodie, ANSI C Chair, to all ANSI C members. To put it in perspective, a member of the public, Mr. Hansberry, submitted a lengthy public comment. Unfortunately, it was mislaid for quite some time and did not surface until all public comment periods had ended and the committee had voted out the standard. Mr. Hansberry's paper was eventually found and the standardization process was put on hold until his issues were addressed. This was done, in his presence, at the Seattle meeting in April 1989. However, Mr. Hansberry found the outcome unacceptable and is attempting to have some of his issues reconsidered. He is also questioning various procedural issues.)

"This is a formal notice that the meeting scheduled for Salt Lake City on September 21 and 22, 1989, has been CANCELLED.

"Let me give you a little background on our current status to help explain why the meeting has been cancelled.

"We submitted the draft standard and our responses to Mr. Hansberry's comments to the X3 Secretariat following the April X3J11 meeting. This was done on the timetable discussed at that meeting.

"Mr. Hansberry, however, did not accept our responses and has written a formal appeal to X3, challenging the work of X3J11. The appeal is fairly lengthy, raising about 40 different technical and procedural issues. It also threatens legal action if he is not satisfied with the results.

"According to the X3 rules only procedural issues can be addressed in an appeal. The technical issues will be considered closed if no negative X3 votes arise during the reconsideration ballot which is currently being conducted. (Mr. Hansberry's letter and our responses were, after a delay because of the appeal, distributed to the X3 members and they have been given an opportunity to change their vote based solely on the issues raised in his paper.) This reconsideration ballot will close August 2, 1989.

"The procedural issues are not so easily addressed. Each appeal issue must be responded to by X3. Tom Plum and Bill Plauger met with the X3 leadership, in Washington, D.C., to help organize, supply information for, and expedite the X3 response to the appeal. The formal procedures for the appeal process, unfortunately, take a considerable amount of time.

"A response to Mr. Hansberry's appeal has been prepared and sent out by X3 (with a lot of help from Tom Plum). The X3 position is that appeal is unwarranted and that no additional remedial action is required.

"If Mr. Hansberry accepts the X3 response (or does not respond by August 15, 1989) and there are no ballot changes during the reconsideration ballot, the proposed draft C standard will be forwarded up to ANSI for consideration during August.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"If Mr. Hansberry does not accept the X3 response to his appeal, then a formal appeal board must be formed. The process for forming and convening this board will probably take a least two months. The findings of this board will govern what further steps must be taken.

"If the board finds in favor of X3J11 (as we currently believe they should), the standard will be forwarded on to ANSI at the conclusion of their work.

"The ANSI committee, BSR, that reviews draft proposed standards meets in the middle of October and the middle of December. If by some outside chance we get the draft proposed C standard to them in time to be considered during the October meeting, we would, if there are no further problems, have a standard by November 1989. If we miss October and make the December meeting, it will probably be January 1990 before we formally have an approved standard.

The bottom line on all of this is that we don't have a C standard yet.

"If still not satisfied, Mr. Hansberry has the option of appealing yet again at the ANSI level. However, in most cases, this appeal is processed after the draft proposed standard receives the American National Standard status.

"The bottom line on all of this is that we don't have a C standard yet. We, therefore, are not in a position to begin the process of doing interpretations. This makes a September meeting of very limited value. Since it seems our time and travel budgets are always strained, the officers believed that it was better to simply cancel this meeting and aim for the March 1990 meeting in New York."

Turn your favorite C compiler into a powerful database manager with the
C/Database Toolchest
The C/Database Toolchest™ adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the C/Database Toolchest™, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.
The C/Database Toolchest™ includes three major components:
1) An advanced B+tree library gives you instant access to your data.
2) A high-level ISAM library provides you with an easy-to-use C interface, and
3) A complete database manager (with C source code included) shows you how to create impressive applications.
You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.
The C/Database Toolchest™ supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.
About the only thing that the C/Database Toolchest™ doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.
Now Only $19.95!
Order Coupon
Name
Street
City
State Zip
Telephone
Paying By Money Order Check
Visa MC AX Disc.
Card#
Exp. Date
Disk Size 5¼" 3½"
Qty. Product Price Subtotal
C/Database Toolchest... $19.95
C/Database Library Source, $10.00 B+tree & ISAM library source code
Add Shipping ($5 USA, $20 Foreign)
Texas Residents Add 8% Sales Tax
Total Amount of Your Order
Order now by calling our toll free number or mail the coupon to:
Mix Software
1132 Commerce Drive
Richardson, TX 75081
1-800-333-0330
60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001
MIX software
C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft C are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.
Number 266 on the Reader Service Card

International Business

In mid 1989, I took on the role of International Representative for the U.S. to the ISO C committee. The previous holder of this office, P. J. Plauger, remains as the ISO C convener. He resigned the IR role since it conflicts with that of convener: The convener's job is to help the group come to a consensus, whereas the IR has a vested interest in a particular position.

Since I have attended three of the last four ISO C meetings, I am not new to the ISO process. If any of you have questions pertaining to international C issues, drop me a line.

Aztec C V4.2

I recently took delivery of Aztec's latest DOS C offering. The compiler is available in several versions. The one I reviewed against my ANSI C compatibility suite was the commercial version.

The Compiler Options

The only option of importance here is *-ansi* which "makes the compiler behave as much as possible like an ANSI compiler," whatever that means.

The Preprocessor

The manual claims that the preprocessor is ANSI conforming; however, I have found this not to be true. The following ANSI C attributes are not supported or had some limitation:

- Produces an error if # appears in the definition of an objectlike macro; it

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windows, Menus, B-trees, Relational Database Libraries

From Marietta Systems

The "c_wndw"™ and "c_ndx"™ libraries extend the facilities of the C language by providing a complete set of window, menu and relational database functions. These libraries are available from Marietta Systems for TurboC, and Microsoft C for PC-DOS and MS-DOS.

The "c_wndw" library:
- o Pull-down and pop-up menus
- o Stacking windows
- o Color and cursor control
- o Editing of numbers and dates
- o Record level file access
- o Full error handling
- o dBASE data dictionaries

The "c_ndx" library:
- o Multiple B-tree indexes
- o Hashed random file access
- o dBASE compatible NDX files
- o Relational file access
- o Multi-level file relations
- o Special currency value fields
- o Database field access

Both object libraries are available for a single price of $95. This includes 114 functions, reference manual, user's guide and sample programs. Source code is available for $195. (S&H $4.50; Georgia residents add sales tax)

Marietta Systems Inc., PO Box 71506, Marietta, GA 30007

(404)565-1560 Visa/MasterCard accepted

Because your time is your most valuable resource.

Number 223 on the Reader Service Card

The C Programmer's Companion

Now Available for OS/2 !!!

CCSYM - C Symbol Compiler

- * Analyzes ANSI C source code. Microsoft/Turbo extensions supported.
- * Automatically generates separate commented function prototype files for all global and local (static) functions.
- * Log File lists all your symbol definitions with detailed usage statistics.
- * Generates a "symbol database" which CCRIDER uses to quickly retrieve ANY symbol definition in your application, no matter how large it is.
- * Incremental compilation supports efficient database updates.

CCRIDER - Pop-up Symbol Lookup

- * Pop-up utility which converts your own editor into a powerful hypertext programming environment.
- * Point at a symbol in your editor, then pop-up its definition in a window.
- * One keystroke "Edit" command allows you to edit the definition in your own editor, no matter which file or directory it was defined in.
- * "Paste" command inserts the definition into your currently edited file.
- * Supports multiple editing sessions under OS/2.

DOS Version $89
OS/2 Version $179

FREE DEMO

Western Wares
Box C Norwood, CO 81423
(303) 327-4898

Number 100 on the Reader Service Card

New Rev2.0 Release & New Products

The C Programmer's Toolbox contains more than 20 tools for the serious C programmer. The tools are now available in three different development environments: PC/MS DOS; Macintosh's MPW *(new)*; and Sun's UNIX *(new)*. Some of the major tools include:

CDecl	Translates C declaration statements to/from English *(helps you decipher those complicated C variable declarations and construct new ones). {MPW & Sun only}*
CFlow™	Determines how a program is organized and what is in the runtime library *(quickly shows you function hierarchy and interdependancies)*
CHilite™	Highlights and prints your C source code *(great for creating your final documentation). {MPW & Sun only}*
CLint™	Syntactically checks one or more of your C source code files *(use this to find those silly, syntactically correct programming mistakes)*
Cpp	Preprocesses C source code files that have ANSI macro definitions *(helps you port ANSI C code to/from your target environment). {MPW & Sun only}*
CPrint™	Reformats/beautifies your C source code *(makes code easy to read and ends disputes on source code formats).*
CritPath™	Determine a program's critical path. *(could help you improve your program's performance) {PC DOS only}*
CXref™	Cross references and checks usage of up to 17 different classes of symbols in your C source code *(through its virtual memory system you can cross reference any size program or number of listings)*
PMon™	Monitors a program's execution *(non-invasively monitors your program's function execution, system service and BIOS usage). {PC DOS only}*

C™ Programmer's Toolbox / MPW PC Sun

The tools include support for ANSI C, Microsoft C, Turbo C, Quick C, Sun UNIX C, Macintosh's MPW C, LightspeedC and more.

This new Toolbox release easily handles any size program through its new virtual memory system. Also, several tools include a new C macro processor that accepts K&R, ANSI and other C dialects.

C Programmer's Toolbox™
MPW ($295), PC Volume I or II ($99.95),
PC Volume I & II ($175), Sun ($495)

C Programmer's McTool Series™ (Macintosh - non MPW)
McCPrint™ ($59.95), McCLint ($99.95)

Call and Order today...
(CA residents, please add sales tax; Overseas shipping extra)

MasterCard
VISA®

MMC AD Systems
Box 360845
Milpitas, California 95035
(408) 263-0781

Number 94 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HIM 2.0

HIM, the Human Interface Manager, is a C library that gives you keyboard handling, windows, menus, context-sensitive help, text editing, and bullet-proof data entry for DOS, UNIX, XENIX, and VMS. Keyboard handling, line drawing, and function calls operate exactly the same across all operating systems with no changes to your code. Of course, complete source code is included at no additional cost.

Windows. Pop-Up, tiled, border of your choice, or none at all. Our wnprintf() function allows color changes in mid-string and handles word wrap automatically.

Menus. Pull-down, Pop-up, or 123-style with nesting to any depth. Size and placement can be automatic, manual, or a combination. Selection of an item automatically calls another menu, a function, or just returns its element number. Menu items can be made not available, have a hot key defined, or tagged by the user for multiple selection lists.

Context-Sensitive Help. Online and only a keystroke away. All help screens are put in a single ASCII file along with a label for each screen. When the user presses the help key (defined via the Keyboard Manager), all text associated with the label appears in a scrollable window. Second press of help key displays keyboard mapping for current terminal.

Keyboard Handling. No more endless switch statements to handle function keys. Define what routine to call when a key is pressed and it's handled for you. Idle background processing lets you display a clock, alarm, or perform any function while waiting for input. Powerful keystroke capturing redirects input to a file for macros or self-running demo playback. Mouse support is enabled with 1 function call.

Data Entry. Full support for all C data types, time, date, money, pop-up lists, and text blocks. Data entry window can be any size you choose, with fields defined on multiple pages. Scrollable fields plus lots of "hooks" make interfacing to your favorite data-base library fast and easy.

Source Code Included. After you register, you get the complete library source code. No extra cost, of course.

Support. The best available. Straight from the HIM developers.

See For Yourself. Call today for our *free* demo disks. Order HIM, and we'll let you try it for a 60-day money back trial period. No royalties, runtime licenses, or runtime modules.

Allsoft
100 Calle Playa Del Sol
Albuquerque, NM 87109
(505) 821-7760

$195 DOS
$395 XENIX
$695 VMS UNIX

Number 322 on the Reader Service Card

should not, since # is only an operator in functionlike macros
- Doesn't diagnose extra tokens at the end of directive lines
- Failed to diagnose an unterminated comment
- Allowed a macro definition to span source files
- Quietly accepts casts in #if directives
- Permits #else directive to be in different file than #if but not #endif

The sizeof operator is permitted in #if arithmetic, although this is not required by ANSI C.

The preprocessor is in very good shape and provides all the new operators and directives. It also carefully follows the defined phases of translation.

The C Language

The following ANSI features are not supported or have problems:

- Permits a const object to be modified
- No auto aggregate initializers permitted
- Accepts long float but should not
- No trigraph support

With the exception of the first, these are very minor issues.

The Run-Time Library

All 15 headers are present; however, only setjmp, signal, and stdarg were completely conforming. All others are incomplete or contained extensions that were not conditionally compiled out in strict ANSI mode. Some contained nonstandard macros and functions, others were missing one of more typedefs, and some had incorrect prototypes.

All of the headers use a protective wrapper to prevent their being #included more than once. However, more protection is needed. For example, size_t is defined in both string.h and stddef.h (and should also have been defined elsewhere). Yet when these headers are included together, the same typedef is seen twice, causing a compilation error. Each such typedef must also have a conditional wrapper.

Numerous function prototypes were missing the const keyword.

Header assert.h cannot be #included more than once with NDEBUG defined and not defined

UNDERSTANDING C VIDEOTAPES

Learn the techniques used by experienced C programmers.

Get guaranteed results from this informative 3-volume, 6-hour videotape course. Loaded with examples that cover every nook and cranny of ANSI C. Handy reference manual included.

$59.95 postpaid (plus tax in Texas). Money-back guarantee. No extra charge for C.O.D. orders.

To order C.O.D., call:
800-762-5515

(Canadian orders: $64.95 in U.S. funds)

AUTOMATION EDUCATION
2309 Royce Drive
Arlington, TX 76016-1232

Number 207 on the Reader Service Card

Parallel Programming for "C"

INTERWORK™
A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX™-style signals. Also has building blocks for constructing your own ITC facilities.
- Lets you trace task switches and inter-task communication.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX™	$159
DEC VAX™, SUN	UNIX 4.2BSD	$249

Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $5.00. Send check or money order to:

Block Island Technologies
Innovative Computer Software
15455 NW Greenbrier Parkway, Suite 210
Beaverton, Oregon 97006
(503) 690-7181 FAX (503) 645-7732

*Trademarks: Interwork, Block Island Technologies; UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

Number 244 on the Reader Service Card

PRODUCED 2001 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

as required. The assert macro is also defined as a statement rather than an expression, prohibiting its use in several legitimate situations.

Header float.h is missing FLT_ROUNDS.

Not surprisingly, the multibyte-character stuff is missing. Also, the locale support is only partial.

Header stddef.h is missing some macros and typedefs and declares errno (an interim location in a previous draft of ANSI C).

The library needs plenty of work.

The Bottom Line

By defining __STDC__ as 1 (and thereby claiming ANSI conformance) Manx invites critical review, and I have accepted the invitation. Despite the fact that the preprocessor and language are very close to ANSI conformance, the library clearly is not. Once again, it's a case of some marketing license.

Despite this, I find the product to be quite solid, particularly when you consider that many of the library problems can be fixed by editing the headers. Aztec customers should feel comfortable that their favorite C development system is keeping in the "mainstream."

Oh—this compiler actually found a bug in my test suite, in the area of function-prototype compatibility. PJ

Rex Jaeschke is an independent computer consultant, author, and lecturer. He is actively involved with the ANSI and ISO C standards committees and is the convener of the Numerical C Extensions Group (NCEG), which is defining numerical extensions to ANSI C. Rex is the editor of The Journal of C Language Translation, *a quarterly publication aimed at C language implementers. If you would like to comment on, or question any aspect of, the ANSI C Standard, contact Rex at 2051 Swans Neck Way, Reston, VA 22091, 703-860-0091 or via uucp: UUNET!AUSSIE!REX. Rex will endeavor to address your concerns in future columns as space permits.*

4c™ *Hypertext* Source Code Databank for Programmers

Explore source code! Navigate uncharted logic trails easily!

4 c includes:

★ **A Source Code Analyzer:** The analyzer maps and cross-references functions and variables to their respective path and file names. The cross-reference table can be displayed alphabetically and by type of variable. It allows a programmer to edit functions directly without specifying a file name.

★ **A Point-and-Shoot Zoom Key:** Which instantly locates any function, global variable, structure definition, #define statement, or macro, etc., in any file, directory, and drive, and displays it in a new edit window.

★ **A Full Featured Built in Editor:** Featuring multiple files, multiple windows, cut-and-paste between files, search-and-replace, auto indent, tab support, and a familiar user interface with pull-down menus, or use your own editor!

"Given a procedure's identifier, an editor should be able to bring up the full procedure interface, including parameters, so that you don't have to remember how many and in what order. Wouldn't it be nice to hit a hot key on the name of a complicated data type and have its definition pop up in a window? It may seem like a real small matter, but how many of you still flip through printouts doing things the editor should be doing?"

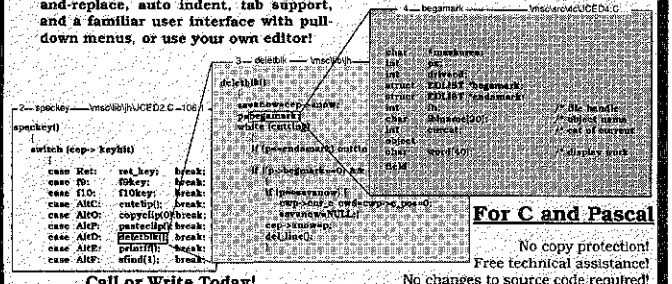
- Phillipe Kahn, "The Connection", Spring/Summer 1988

4 c does this and more

"4c's Magic is apparent..."
- PC Magazine, February 1989

"Daylight in the C Confusion"
- Toolbox magazine, December 1988

"...Destined to become the standard editor for C..."
- Computer Persontich magazine, October 1988



For C and Pascal

No copy protection!
Free technical assistance!
No changes to source code required!
Required IBM PC compatible with 350K RAM.
30 Day Money Back Guarantee!
Only $119.00 plus shipping
Visa, Mastercard, check, money order, or COD.

Call or Write Today!

Tri-Technology Systems Inc.
1225 S. Elgin, Forest Park, Il. 60130
Phone: 312-366-7595

Number **276** on the Reader Service Card

PC-lint *The # 1 Lint for MS-DOS*

```
        void
      main() (
   struct(unsigned
  a:2, b:3, c:4;)s;
 unsigned row,col;char
ch;int x;char y;unsigned
z;void          printf ( )
; /*             a lie but
                  harmless
                  */ x=0 ;
                   /* x is
                    set  to
                   zero*/row
                   =25;col=
                  80; if (
                 row+col>
             -1)x++;ch=
          '\xFF';if(
         (unsigned)
    ch==0xFF)
   x++;z=2;
   y=(z &
   0x3)<<
   7; if(
   y)x++;
   z =12;


     if(z/
  -1)x++;s.b
  =10;if( s.b
  >5) x++;z =
  37;if(-z<0)
  x++;printf(
   "x=%d\n",
    x );)
```

You have just inherited this C program.

What do you do?

(a) Run away

(b) Run PC-lint

Use PC-lint to examine foreign looking C code and report on potential problems and difficulties. PC-lint's many options let you quickly and easily suppress unwanted error messages and extract just the information you need.

PC-lint is a C source code analysis tool. It will check code more thoroughly than a C compiler and, because it looks across multiple modules, it enjoys a perspective that your compiler doesn't have.

Control your development time and cost. Locate source code errors that could otherwise go undetected for days, months or even years.

Over **200 error messages**. More than **70 options** for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, both, etc. **Easy to Use** - If you know C you already know how to use PC-lint. Check for **portability** problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI **prototypes** for your K&R functions.

Mainframe and Mini Programmers

Get **FlexeLint** (formerly Generic Lint) in shrouded source form for VAX/VMS, Unix, DG AOS/VS, OS-9, Apple MPW, IBM VM/MVS, etc. Requires only K&R C to compile but supports ANSI. Pricing starts at $798. Call for details.

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426

(215)584-4261

Only $139 - CALL TODAY

30 Day Money-back Guarantee

Specify MS-DOS or OS/2. Quantity and educational discounts available. PA add 6% sales tax -- Outside USA add $20. PC-lint and FlexeLint are trademarks of Gimpel Software.

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Using OOP to Create a Powerful File Viewer

by Tom Swan

"There is not a fiercer hell than the failure in a great object."
—John Keats

You've probably heard that object-oriented programming promises to help you write code that's easy to extend with a minimum of fuss and bother. But will OOP's great promise be kept, or will it turn up empty? To find out, I've started a project that I hope will also solve one of my pet peeves: having to use umpteen different programs to view the contents of files containing anything from ASCII text to graphics and spreadsheets.

Leaning heavily on Turbo Pascal 5.5's new OOP extensions, the finished program—named VA, for "View Anything"—will readily adopt new file-type definitions. For example, it will be a simple matter to add support for a custom graphics device simply by defining that device's characteristics; no other modules will require modifications. Eventually, I hope to publish all of the source code in *PJ* and to invite contributions from readers. Along the way, I'll share my observations, praise, and criticisms about OOP customs.

I'll begin in this issue and the next with a harvest of several low-level modules from an early VA prototype. Together, the pieces form a modular menu-command system that allows actions to insert themselves into a program's menu tree—a good example of OOP and *polymorphism*, one of OOP's many buzzwords that I'll be picking on from time to time.

Top Down or Bottom Up?

A commonly asked question is "Should OOP code be designed in top-down or bottom-up fashion?" More so than ever, I'm finding that OOP demands a rigorous top-down strategy. I can say that with authority because of all the time I've wasted beating around the low-level OOP bushes. While learning OOP's ropes, I wrote armloads of wonderful video, keyboard, and list objects for which I've yet to find any good use.

Only by starting at the top with a well-defined goal—in this case, my VA program—was I able to begin designing low-level objects of practical purpose. Having a goal also helped me to remember an obvious but easily forgotten rule: Everything in a TP 5.5 program doesn't have to be object-oriented!

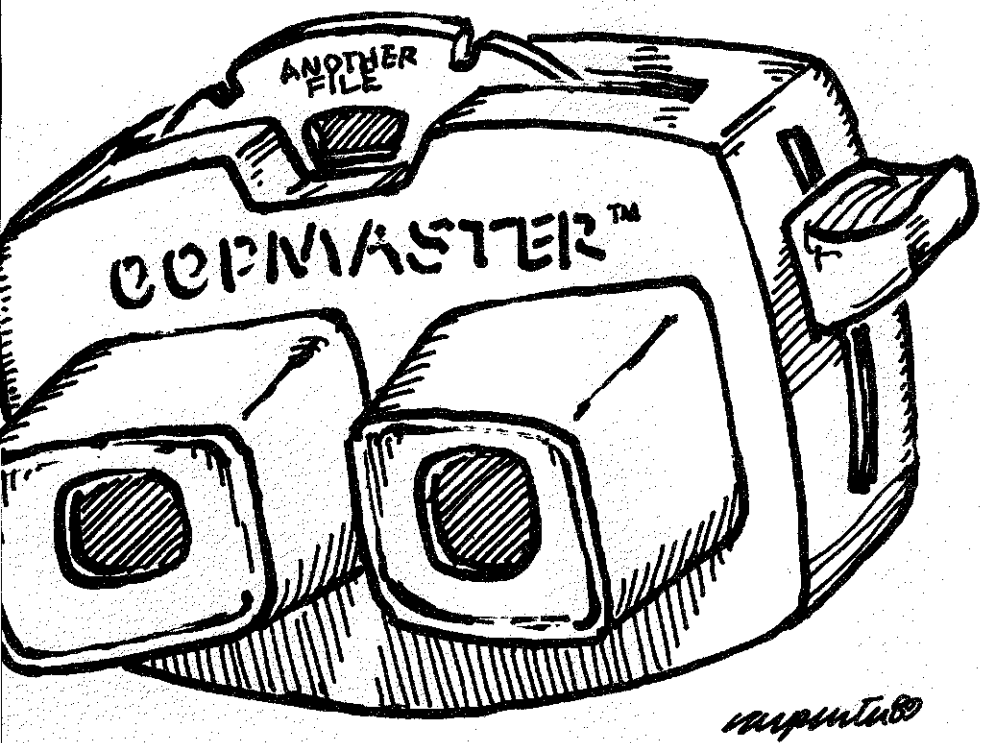
For example, ukeys.pas (Listing 1) has no objects at all, and it probably needs none. This simple module contains two functions and one procedure: **getKey** returns a character from the keyboard, **keyWaiting** returns true if a character is waiting to be read, and **pushKey** inserts one character back into ukeys, which the next call to **getKey** will return. This lets me simulate keypresses with statements such as:

While learning OOP's ropes, I wrote armloads of wonderful video, keyboard, and list objects for which I've yet to find any good use.

```
pushKey( chr(27) ); { simulate <Esc> keypress }
```

The ukeys module has two limitations: It doesn't handle function keys, and **pushKey** can store only one

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



character at a time. If you pass two characters to **pushKey** without calling **getKey** in between, the first character is lost. These limitations are temporary, and I plan to redesign the module eventually.

Lists in the Abstract

Objects and lists go together like tortillas and beans—and for good reason: Lists are essential for writing extensible OOP code. By designing objects that can attach themselves onto lists, descendant objects can insert new commands into menus and inject custom shapes into CADD drawing libraries. Other code can traverse a program's lists to activate commands and draw pictures.

Before designing the **list** object, we first need to consider what the list will contain. However, in creating a truly general list manager that can link up any kind of object, it's impossible to predict every detail. The typical OOP solution to this Catch-22 is to define an *abstract object*, which is never instantiated (declared as a variable). Abstract objects serve only as grandparents for future descendants, in this case, the objects that eventually will be stored in lists.

Module uitem.pas (Listing 2) defines the abstract object **item**. Following typical OOP conventions, the complete definition includes a pointer and an object type:

```
itemPtr = ^item;
item = object
    left, right : itemPtr;
    constructor init;
    destructor done; virtual;
    procedure link( ip : itemPtr );
    procedure unlink;
    procedure processItem; virtual;
end;
```

Type **itemPtr** addresses objects of type **item**. Inside **item** are two **itemPtr** instance variables (or fields), *left* and *right*. A simple constructor **init** and destructor **done** initialize and deinitialize **item** objects. The constructor contains only two statements:

```
left := @self;
right := @self;
```

The normally hidden *self* parameter, which is passed to all methods, has the same data type as the object that defines the methods. With an at-sign (@) prefix, *@self* returns the address of the object instance that activated the method; therefore, the two assignments set *left* and *right* to the address of that object. I've designed **item** this way to have a clear definition for an object that's not linked to any other: If an object's *left* and *right* fields equal *@self*, then we know that the object is not a member of any list.

The destructor **done** has no statements, but it's not a do-nothing shell. When called via **dispose**, an empty **done** still serves to dispose any heap memory allocated to the object by a previous call to **new**.

In addition to the constructor and destructor, **Item** also defines three methods:

- **link** — links an item object to another at *ip*
- **unlink** — unlinks an item object from another
- **processitem** — performs an undefined action

Two of the methods allow **Item** objects to link and unlink themselves from other items, adding the native ability for **item** descendants to form lists. Method **processItem** has no defined purpose as yet. Objects descended from **item** are expected to redefine **processItem** appropriately. For example, in a graphics object, **processItem** might draw a shape. In the menu-command object to be described next time, **processItem** will perform a command's action.

To provide such versatility, **processItem** must be *virtual*—a word that triggers a raging controversy in OOP circles. In Turbo Pascal, methods ending with the keyword **virtual** are addressed by way of a lookup table called the virtual method table (VMT). There is one VMT stored in the global data segment for each object *type* that defines a constructor or has one or more virtual

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

methods. The object instances do not contain VMTs—a common misconception.

The Static or Virtual Controversy

Because virtual-method calls are made by looking up subroutine addresses in the VMT, descendant objects (which add their own VMTs to the data segment) modify a program's behavior by redirecting such calls to their own replacement methods. The compiler handles the details of making this happen—all you have to do is use the **virtual** keyword and be sure to call the object's constructor to link the object instance at run time to its VMT. The redirection of virtual-method calls goes by the fancy name *polymorphism*—the ability for descendant objects to redefine the run-time behavior of existing code.

Static methods (those that do not end in **virtual**) are called just like other procedures and functions. Because a static method's address is hard-wired into the code, unmodified statements in ancestor objects can't call new static methods defined by object descendants.

So, which method variety—static or virtual—is better? That's the $64,000 question. In one early VA version, I defined *every* method as virtual—until I realized how silly that was. The techniques for linking and unlinking listed items are unlikely to require future modification; therefore, I decided to make **link** and **unlink** static. On the other hand, **processItem** must be virtual for future descendants to redefine what this method does.

I'm more than a little bugged by having to make such decisions. Some people avoid the question by making all methods virtual, as I had done originally. However, because I want to explore all Turbo Pascal OOP techniques, I don't want to restrict myself to a narrow point of view. So, I've decided to mix static and virtual methods. I just hope I'm not shooting myself in the foot.

From Items to Lists

To prepare for writing the next module, ulist.pas (Listing 3), I examined several other OOP-list implementations, and I made a surprising discovery: In most cases, lists and the items stored in lists were defined as separate object types. It seems to me that a **list** object ought to be descended from the abstract **item**. Lists are naturally recursive (especially ordered lists), and this plan lets me create lists of lists, which might come in handy for designing sparse arrays and multilevel command menus. Here's the result:

```
listPtr = ^list;
list = object( item )
    anchor : itemPtr;     { Addresses list head }
    cip : itemPtr;        { Current item pointer }
    constructor init;
    destructor done; virtual;
    function listEmpty : Boolean;
    function atHeadOfList : Boolean;
    function atEndOfList : Boolean;
    function currentItem : itemPtr;
    procedure prevItem;
    procedure nextItem;
    procedure resetList;
    procedure insertItem( ip : itemPtr ); virtual;
    procedure removeItem( ip : itemPtr ); virtual;
    procedure processItems; virtual;
    procedure disposeList; virtual;
end;
```

List inherits the properties of **item** by specifying the object's ancestor in parentheses after the **object** keyword. An instance variable *anchor* points to the head of the list and never changes. Another variable—*cip*, for "current item pointer"—floats around, addressing any listed item. The **init** constructor's implementation contains the following statements:

```
anchor := nil;
cip := nil;
item.init;
```

After setting *anchor* and *cip* to nil, **init** calls the ancestor's constructor to finish the initialization. Chaining constructors together this way is standard OOP protocol. Consequently, because **item** objects initialize themselves by setting their *left* and *right* fields to *@self*, in the descendant **list** object, the result is to initialize an empty list.

List's destructor **done** is equally simple, calling **disposeList** if the **list** is not empty and then passing control to the ancestor's destructor:

```
if anchor <> nil
    then disposeList;
item.done;
```

The **list** object also defines three Boolean functions with obvious meanings—**listEmpty**, **atHeadOfList**, and **atEndOfList**. Another function, **currentItem**, returns the value of *cip*. Some people recommend *never* directly accessing object fields such as *cip* outside of the object's own methods; therefore, functions like **currentItem** are commonplace in OOP. I'm not yet convinced of this rule's importance, but I'll try anything once, and so **currentItem** stays—at least for now.

Two other procedure methods move *cip* to the previous (**prevItem**) and next (**nextItem**) objects. **ResetList** resets *cip* to address the first item in a list. Like many OOP list objects, mine are circular, but they still have a head and a tail.

Notice that except for the **done** destructor, all methods so far are static. I assume that none will require future modifications by descendants. Hope I'm right.

I'm not as indecisive about the four final **list** methods. These *must* be virtual to allow replacing by descendants. **InsertItem** inserts **item** object instances into a list. **RemoveItem** removes one item from a list. **ProcessItems** (plural) calls **item.processItem** (singular) for each object in a list. (Perhaps naming these two methods so similarly was a mistake, but it has not been a problem so far.)

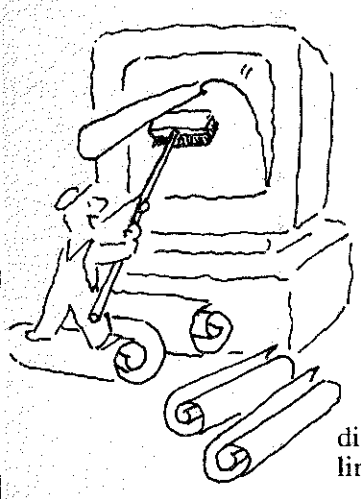
Take a look at the implementation for **processItems** in ulist.pas. First, **resetList** makes sure that *cip* addresses the first listed item. Then, a **repeat** loop calls **currentItem^.processItem** and **nextItem** until the current item again addresses the list head. In this way, **list.processItems** calls the **processItem** method for each listed object in the order in which the items were inserted into the list. Most important, future descendant objects can define their own **processItem** virtual methods, which **list.processItems** will call from inside the previously compiled **repeat** loop—another good example of polymorphism in action.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Two great tools.

NEW! SUPPORTS TURBO PASCAL 4.0, 5.0 & 5.5 AND QUICKPASCAL!

SAYWHAT?! The lightning-fast screen generator.



Whether you're a novice programmer longing for simplicity, or a seasoned pro searching for higher productivity, you owe it to yourself to check out Saywhat. You see, with Saywhat, you can build beautiful, elaborate, color-coded screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can be displayed with as little as one line of code in *any* language.

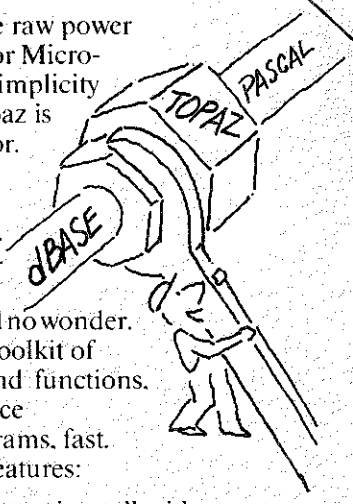
Here's what you get:

- Design screens, windows, and *moving bar menus!*
- Easy-to-use, powerful editor lets you create screens in a jiffy.
- Pop up your screens and menus with one line of code in dBASE, all the dBASE compilers, your favorite BASIC, Pascal, *or any other language!*
- Screen Library Manager.
- Generates runtime code.
- No runtime license or royalty fees.
- Comes with a 100 page manual, plus dozens of sample programs and free utilities.

$49.95

TOPAZ The breakthrough DBMS toolkit for Pascal

If you'd like to combine the raw power and speed of Turbo Pascal or Microsoft's QuickPascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. That's because Topaz was specially created to let you enjoy the best of *both* worlds. The result? You create complete, truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you produce outstanding, polished programs, fast. Check out these powerful features:



- Over 200 routines all with easy-to-use, dBASE-like syntax.
- Data entry routines like **SAY, GET, PICTURE, RANGE,** color selection, unlimited data validation.
- Open up to 10 DBF files, with up to 7 indexes with **USE, SELECT, SKIP, APPEND, PACK, INDEX ON, SET INDEX TO,** and **FIND.**
- No need to buy dBASE. **CREATE, BROWSE** and **REPORT** utilities included.
- Easily implement Saywhat and Lotus-style moving bar menus.
- **BROWSE** any DBF file with just one line of code! Programmable and windowed too.
- Pick from windowed data or filenames with one line of code.
- Comprehensive Time & Date math in 7 international formats.
- Powerful code and report generators included!
- Comes with a complete 250 page manual, plus sample programs to get you started.

$74.95

ORDER NOW. YOU RISK NOTHING.

Thousands of satisfied customers have already ordered from us. Why not visit your dealer or call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

Special limited-time offer! Save $26. Buy Saywhat?! and Topaz together for just **$99** (plus $5 shipping and handling).

Visit your nearest dealer or call toll-free:

800-468-9273

In California: 800-231-7849
International: 415-571-5019

Software Science, Inc.
The Research Group
100 Valley Drive, Brisbane, CA 94005

MONEY BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, for any reason, return them within 30 days for a prompt, friendly refund.

Dealers: SAYWHAT?! and TOPAZ are available from Kenfil Distribution, and in Europe from ComFood Software, W. Germany 49-2534-7093

Guaranteed!

SOFTWARE SCIENCE INC.

Number 198 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The virtual **disposeList**, which is called by the **done** destructor, operates similarly to **processItems**, but it stores **currentItem** in a temporary *ip* variable of type **itemPtr**. It would be a mistake to write

```
while not listEmpty do
begin
    removeItem( currentItem );  { Yikes!!! }
{ ... }
end;
```

Instead, the listed object must first be unlinked from the list and then disposed, thereby recovering its memory for future use. An **if** statement handles the disposal details:

```
if ( seg( ip^ ) <> DSeg ) and ( seg( ip^ ) <> SSeg )
    then dispose( ip, done )
    else ip^.done;
```

Most OOP list managers I've seen just call **dispose**, assuming that all objects were allocated space on the heap by **new**. In my design, **disposeList** checks whether an object is in the data segment or is a local variable on the stack. In either event, **disposeList** directly calls the virtual **ip^.done**, which presumably cleans up any internal affairs inside the object instance. Only if the object is in heap memory does the program call **dispose**, passing the destructor **done** as the second parameter. This cleans up the object and recovers any heap space allocated by **new**.

String Items

The third and final module for this installment, ustritem.pas (Listing 4), defines another **item** descendant. As usual, the module specifies a pointer and object type:

```
strItemPtr = ^strItem;
strItem = object( item )
    sp : ^string;
    constructor init( s : string );
    destructor done; virtual;
    function getString : string;
    function putString( s : string ) : Boolean;
            virtual;
    procedure upperString; virtual;
end;
```

I want **strItem** objects to use memory efficiently; so, instead of using a fixed-length variable, I defined a pointer **sp** to address a **string** variable allocated on the heap. With this design, **string** objects take up only as much memory as they need.

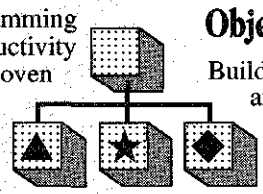
The constructor **init** requires some explanation. It contains the following statements:

```
sp := nil;
if putstring( s ) then item.init else
begin
   done;
   fail;
end;
```

After initializing **sp** to nil, **putString** attempts to store the constructor's string parameter *s* on the heap. If this

The Professional Library for Object-Oriented Pascal

Object-oriented programming multiplies your productivity by promoting reuse of proven software modules. Don't start from scratch! Use **Object Professional 1.0**, a powerful library of over 30 object types containing over 1000 methods.



Powerful User Interfaces

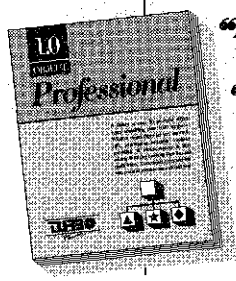
The window object types let you use multiple overlapping and resizeable windows. The windows provide capability for ■ mouse support ■ scroll bars ■ menus ■ text editing ■ dialog boxes ■ pick lists ■ scrolling data entry screens ■ printed forms ■ help capability, and more. The window classes are incredibly flexible — you can create text-mode PM look-alikes or your own unique look and feel.

Object Oriented Data Too

Build your programs from proven and documented object types like stacks, linked lists, virtual arrays, and more. Make your own custom data types by simply inheriting from one of the provided types and adding your own methods and instance variables.

The Impossible, Made Easy

System-oriented routines provide ■ swappable TSRs in only 6K of RAM ■ DOS and BIOS capabilities ■ EMS management ■ keyboard macros ■ interrupt management ■ swapping Exec manager, and much more.



Complete Documentation, Full Source

Object Professional has three volumes of complete documentation, online reference guide, free technical support, and full source code. You pay no royalties. Hot demo programs show you how to use the power of Object Professional. You'll get up to speed fast with OOP!

Object Professional is the successor to the acclaimed Turbo Professional:

"A superbly crafted toolbox."
Kent Porter, DDJ, 4/88

"The range of this toolkit is simply astonishing."
Jeff Duntemann, DDJ, 5/89

Object Professional 1.0, only $150.

Call toll-free to order.

1-800-333-4160

8AM - 5PM PST Monday through Friday, USA & Canada.
For more information call (408) 438-8608.
TurboPower Software PO Box 66747 Scotts Valley, CA 95066-0747

Satisfaction guaranteed or your money back within 30 days. Turbo Pascal 5.5 is required. Add $5 for shipping in U.S. and Canada. Elsewhere add $35 per unit. Turbo Professional 5.0 customers may upgrade for $60, include your serial number.

TURBO Power

Number 25 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

succeeds, the object continues the initialization by calling **item.init**, completing the constructor chain.

If **putString** fails, the constructor calls **done** and **fail**—a simple yet effective error-handling strategy. Calling **done** ensures that any partial object structures are disposed. (There aren't any in this case, but it's usually wise to call an object's destructor if the constructor ends with an error.) **Fail** immediately exits the constructor (similar to the way **exit** works) and also deallocates any heap space allocated to the object under construction. To avoid a run-time out-of-memory error, the program must install a custom **heapFunc** error handler—a topic I'll cover another time. In that case, **fail** causes **new** to return a nil pointer, a condition the caller can check.

In method **putString's** implementation, **getMem** and **putMem** allocate and dispose space for string variables on the heap. When calculating how much space to reserve for a string, it's necessary to add 1 to the string length to account for the length byte that prefaces all Turbo Pascal string variables.

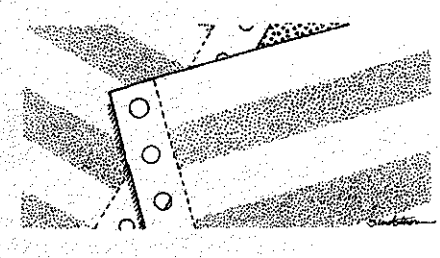
As an afterthought, I also threw in method **upperString** to convert strings to uppercase. I'm always rewriting this little procedure, and so it seemed reasonable to include it in **strItem**.

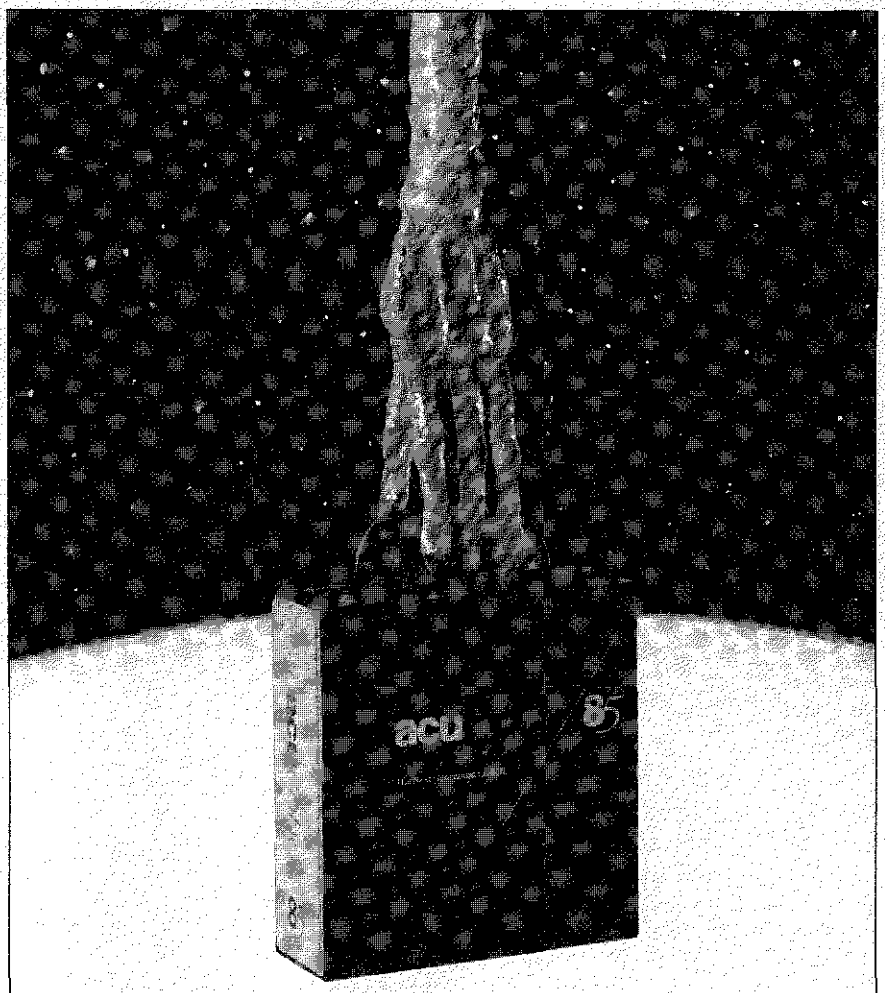
More to Come

In the next installment, I'll present the remaining modules that implement the self-installing menu-command system. Meanwhile, you may want to compare my list implementation with others. You can send all comments to me in care of *PJ*, or send me an EasyPlex message via CompuServe (73627,3241). PJ

Tom Swan is a contributing editor of Programmer's Journal *and the author of* Mastering Turbo Pascal 5.5 *and* Mastering Turbo Assembler, *both published in 1989 by Howard W. Sams, Indianapolis, Indiana.*

Code follows.





INTRODUCING
THE MOST PORTABLE COBOL
IN THE UNIVERSE

Go where you want to go
Transport COBOL programs to almost any platform—without changing one line of code. Run on AIX, MS-DOS, NETBIOS, UNIX, OS/2, ULTRIX, XENIX and VMS without recompiling. No other COBOL-85 offers such hardware independence, or the instantly-expanded possibilities.

Open new windows in your applications
Create better-looking programs with windows. Add help screens, pop-up/pull-down menus, context-sensitive messages, more. ACUCOBOL-85 has a Display Window verb, including extensions for boxed, reverse-video and pop-up windows in color. Paint special forms on the screen with our Display Line verb. And scroll up/down inside windows.

Don't get lost in space:
ACUCOBOL-85 keeps you on course
ACUCOBOL-85 is NBS certifiable error-free under the High 1974 and Intermediate 1985 ANSI standards. If you lose course, it has its own interactive source code debugger that's fast and easy to use.

Step into the future
Make your job simpler and stay competitive—even ahead of the game. Put the next generation COBOL to work for you today.

Free evaluation copy
Call (619) 271-7097 today for your free evaluation copy of ACUCOBOL-85. And see the shape of things to come.

acucobol inc

Voice: (619) 271-7097 Fax: (619) 566-3071
7950 Silverton Avenue Suite 201 San Diego California 92126

ACUCOBOL-85/the ACUCOBOL logo are trademarks of ACUCOBOL, Incorporated; MS-DOS/XENIX of Microsoft Corporation; NETBIOS/OS/2/AIX of International Business Machines Corporation; VMS/VAX/ULTRIX of Digital Equipment Corporation; UNIX of AT&T Bell Laboratories; RM/COBOL and RM/COBOL-85 of Ryan-McFarland Corporation.

Number 282 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

No Other MS®-DOS Linker...

Helps you overcome the 640K barrier with a virtually unlimited and flexible overlay manager,

and....

helps you improve program execution speed,

and....

overcomes the large program "compiler fast but linker slow" problem

and....

prevents duplicate code in multiple EXE systems from chewing up disk space on development machine, shipping media, and target machine,

and....

provides in-depth, highly detailed information on interrelationships and dependencies of all modules of your program,

and....

works with almost all MS-DOS compilers.

Powerful Overlay Manager

RTLink™ supports two separate overlay schemes.

The first of these schemes parallels the method used in the Microsoft® linker. It supports one overlay area and must have language library support.

The second overlay scheme which **RTLink** provides is much more flexible. It allows up to 64,000 overlay areas to be interspersed at separated or contiguous locations within the memory image. Each overlay area can in turn contain smaller, nested overlay areas within it which can differ depending on which primary overlay is in memory at the time. The portions of code which are brought into each overlay area (called overlay sections) can be placed into either the executable (.EXE) file or into separate, external files whose names you specify. There can be up to 64,000 overlay sections. You can specify that a specific overlay section be loaded into a specific overlay area when the program is brought into memory during program startup. You can specify the code which is to be a part of a specific overlay section by naming either object files, modules, classes, or groups. There is no overlay interrupt to conflict with the other interrupts on an end user's system. This second scheme requires no language library support and is language independent. It represents the state-of-the-art in overlay technology.

Fast Development Linking

Run-Time Libraries (.RTLs) are optional, easily-created files that can be used to manipulate link speeds. During development, RTLs can be used to store pre-linked modules. This allows programmers to set aside much of the code that is not being modified, thereby significantly enhancing link speeds. In straight links, **RTLink** is already faster than your current overlay linker. However, **RTLink** is even faster when RTLs are used as a productivity tool during the development and testing phase.

Profiling

RTLink comes with a built-in performance analyzer. This profiler provides highly-detailed performance statistics about your program. Unlike other performance analyzers, these statistics are obtainable without code modification at user-determined/machine-capable timing sequences adjustable to thousandths of a second. Source code is provided for a read/analysis utility which can be used to customize the statistics reports. The statistics provided establish where the time was spent during the program execution and what calls were made as the program executed.

Significant Disk Space Savings

RTLs can significantly reduce the amount of disk space needed to store and/or ship your programs if you have multiple EXE files. When multiple EXEs are present, much of the code (especially language library and certain utility code) is commonly used in each EXE. By placing this redundant code and/or data into a RTL(s) that can be shared by each EXE, you eliminate duplicate storage - which saves disk space.

Interrelationships of Program Modules

The LIBLIST utility provided with **RTLink** can be used to obtain in-depth, highly-detailed information about the interrelationships and dependencies among all modules of a program. This information is useful in overlay design as well as creation and maintenance of run-time libraries.

- **Works with almost all MS-DOS compilers.**
- **No additional memory utilized**
- **No source code changes**
- **No charge for Tech Support**

TO ORDER: Phone: 1-800-456-7032 Fax: 1-713-460-2651

OEM, Multiple-User Licenses, or Technical Information, please call:
1-713-460-5600

.RTLink
Price $195.00

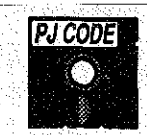
Pocket Soft, Inc.
7676 Hillmont, Suite 195
Houston, Texas 77040
(713) 460-5600

NOTE: Texas Residents include 7% sales tax.

Trademarks: MS and Microsoft are registered trademarks of Microsoft Corporation. .RTLink is a trademark of Pocket Soft, Inc. (c) 1989, Pocket Soft, Inc.

Number 279 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from ". . . a Powerful File Viewer," by Tom Swan

1

LISTING 1

```
(* ukeys.pas -- (c) 1989 by Tom Swan *)

unit ukeys;

interface

uses crt;

function getKey : char;
function keyWaiting : Boolean;
procedure pushKey( ch : char );

implementation

const

   NULL = #0;                  { ASCII NULL character }
   pushedChar : char = NULL;   { Saved char from pushKey procedure }

{ ----- Return next keyboard or pushed character. }

function getKey : char;
begin
   if pushedChar <> NULL then
   begin
      getKey := pushedChar;
      pushedChar := NULL;
   end else
      getKey := ReadKey;
end;

{ ----- Return true if a character is waiting to be read. }

function keyWaiting : Boolean;
begin
   keyWaiting := keypressed or ( pushedChar <> NULL );
end; { keyWaiting }

{ ----- Push a character back "into" the keyboard. }

procedure pushKey( ch : char );
begin
   pushedChar := ch;
end;

end.
```

LISTING 2

```
(* uitem.pas -- (c) 1989 by Tom Swan *)

unit uitem;

interface

type

   itemPtr = ^item;
   item = object
      left, right : itemPtr;
      constructor init;
      destructor done; virtual;
      procedure link( ip : itemPtr );
      procedure unlink;
      procedure processItem; virtual;
   end;
```

2

```
implementation

{ ----- Initialize a new item object to point to itself. }

constructor item.init;
begin
   left := @self;
   right := @self;
end;

{ ----- Dispose item object's memory. }

destructor item.done;
begin
end;

{ ----- Link item to another item addressed by ip. }

procedure item.Link( ip : itemPtr );
begin
   right := ip;
   left := ip^.left;
   ip^.left^.right := @self;
   ip^.left := @self;
end;

{ ----- Unlink an item if it's linked to another. }

procedure item.Unlink;
begin
   left^.right := right;
   right^.left := left;
end;

{ ----- Process contents of item. }

procedure item.processItem;
begin
end;

end.
```

LISTING 3

```
(* ulist.pas -- (c) 1989 by Tom Swan *)

unit ulist;

interface

uses uitem;

type

   listPtr = ^list;
   list = object( item )
      anchor : itemPtr;     { Addresses list head }
      cip : itemPtr;        { Current item pointer }
      constructor init;
      destructor done; virtual;
      function listEmpty : Boolean;
      function atHeadOfList : Boolean;
      function atEndOfList : Boolean;
      function currentItem : itemPtr;
      procedure prevItem;
      procedure nextItem;
      procedure resetList;
      procedure insertItem( ip : itemPtr ); virtual;
      procedure removeItem( ip : itemPtr ); virtual;
      procedure processItems; virtual;
      procedure disposeList; virtual;
   end;
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Swan code, continued

3

```
implementation

{ ----- Initialize an empty list. }

constructor list.init;
begin
   anchor := nil;
   cip := nil;
   item.init;
end;

{ ----- Dispose any listed items and the list object itself. }

destructor list.done;
begin
   if anchor <> nil
      then disposeList;
   item.done;
end;

{ ----- Return true if list is empty. }

function list.listEmpty : Boolean;
begin
   listEmpty := ( anchor = nil );
end;

{ ----- Return true if current item is at the head of the list. }

function list.atHeadOfList : Boolean;
begin
   atHeadOfList := ( anchor <> nil ) and ( cip = anchor );
end;

{ ----- Return true if current item is at the end of the list. }

function list.atEndOfList : Boolean;
begin
   atEndOfList := ( anchor <> nil ) and ( cip = anchor^.left );
end;

{ ----- Return item addressed by current item pointer (cip). }

function list.currentItem : itemPtr;
begin
   currentItem := cip;
end;

{ ----- Move current pointer to previous item in list. }

procedure list.prevItem;
begin
   if cip <> nil
      then cip := cip^.left;
end;

{ ----- Move current pointer to next item in list. }

procedure list.nextItem;
begin
   if cip <> nil
      then cip := cip^.right;
end;

{ ----- Reset list. currentItem will return first item inserted. }

procedure list.resetList;
begin
   cip := anchor;
end;

{ ----- Insert item addressed by ip ahead of current item. }
```

4

```
procedure list.insertItem( ip : itemPtr );
begin
   if ip <> nil then          { Prevent out-of-memory disasters }
   if anchor = nil then       { If list is empty ... }
   begin
      anchor := ip;           {  then start a new list }
      resetList;              {  and initialize current item }
   end else
      ip^.link( cip );        {  else link item into list at cip }
end;

{ ----- Remove listed item addressed by ip and adjust anchor if
necessary to make sure that anchor and cip don't address the removed
item. }

procedure list.removeItem( ip : itemPtr );
begin
   if ip^.right = ip then  { If only one list item ... }
   begin
      anchor := nil;       {  then empty the list }
      cip := nil;
   end else                {  else adjust anchor and cip }
   begin
      if ip = anchor
         then anchor := anchor^.right;
      if cip = ip
         then cip := cip^.right;
   end;
   ip^.unlink;
end;

{ ----- Process all listed items. }

procedure list.processItems;
begin
   resetList;
   if currentItem <> nil then
   repeat
      currentItem^.processItem;
      nextItem;
   until atHeadOfList;
end;

{ ----- Dispose items in a list. }

procedure list.disposeList;
var
   ip : itemPtr;
begin
   while not listEmpty do
   begin
      ip := currentItem;
      removeItem( ip );
      if ( seg( ip^ ) <> DSeg ) and ( seg( ip^ ) <> SSeg )
         then dispose( ip, done )
         else ip^.done;
   end; { while }
end;

end.
```

LISTING 4

```
(* ustritem.pas -- (c) 1989 by Tom Swan *)

unit ustritem;

interface

uses uitem;

type
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
strItemPtr = ^strItem;
strItem = object( item )
  sp : ^string;
  constructor init( s : string );
  destructor done; virtual;
  function getString : string;
  function putString( s : string ) : Boolean; virtual;
  procedure upperString; virtual;
end;

implementation

{ ----- Initialize a new string item, allocating heap space for the
string's characters and addressing that space with the object's sp
pointer. }

constructor strItem.init( s : string );
begin
  sp := nil;
  if putstring( s ) then item.init else
  begin
    done;    { Clean up any partially allocated structures }
    fail;    { Out of memory error }
  end;
end;

{ ----- Dispose of the allocated heap space and any memory assigned
to the string item object. }

destructor strItem.done;
begin
  if sp <> nil
    then freemem( sp, length( sp^ ) + 1 );
  item.done;
end;

{ ----- Return the value of the string. }

function strItem.getString : string;
begin
  getString := sp^;
end;

{ ----- Assign or replace the object's character string. Disposes any
current string and returns true for success; false if out of memory. }

function strItem.putString( s : string ) : Boolean;
begin
  if sp <> nil
    then freemem( sp, length( sp^ ) + 1 );
  getmem( sp, length( s ) + 1 );
  if sp = nil then putString := false else
  begin
    sp^ := s;
    putString := true;
  end;
end;

{ ----- Convert string to all uppercase. }

procedure strItem.upperString;
var
  i : integer;
begin
  if sp <> nil then
    for i := 1 to length( sp^ ) do
      sp^[ i ] := upcase( sp^[ i ] );
end;

end.
```


TechnoJock's Turbo Toolkit

Attention Turbo Pascal programmers.....

Cut your program development time in half and create sophisticated user interfaces in minutes with *TechnoJock's Turbo Toolkit!*

The *Toolkit* contains twelve units with all source code included, and a 300 page manual full of examples. These range from a comprehensive unit that provides a Sidekick Plus style directory window with all the expected sorting features, to a basic procedure that writes text centered on the screen.

And it's easy to use! If you've bought other toolkits but didn't use them because of all their complicated rules ... TTT is for you.

TECHNOJOCK SOFTWARE

- **ultra fast screen writing**
- **EGA/VGA extended lines**
- **exploding boxes**
- **virtual screens**
- **sorted directory listings**
- **comprehensive form input**
- **pulldown menus**
- **nested menus**
- **string manipulation**
- **windows**
- **full mouse support**
- **keyboard control**
- **date manipulation**

"TechnoJock's Turbo Toolkit may just cut your work in half! It provides easy-to-implement procedures.... designed for all Turbo Pascal programmers. With this Toolkit everyone becomes a TechnoJock."

Turbo Users Group

"Here, despite the whimsical name, is shareware at its best: an innovative product, a superb manual, quick mail-order service, and a reasonable price... If you're programming in Turbo Pascal, you absolutely must have TTT!"

Public-Domain Software & Shareware 2nd edition

"Looks great! We have decided to implement your Toolkit in in-house programs for Tech Support."

Rich Kubota
Technical Support
Borland International

TechnoJock Software, Inc.
PO Box 820927
Houston, TX 77282-0927

Orders: 1-800-242-4775
Enquiries: (713) 493-6354

Only $49.95!

Plus $5.00 shipping and handling (Canada $10.00, overseas $15.00). Texas residents add 8% Sales Tax.

Send checks to TechnoJock Software. Send Visa/Mastercard details to: Public (Software) Library PO Box 35705 Houston TX 77232-5705

Number **289** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C THE LIMIT!

MicroWay is your best source for the software and hardware you need to get true 32-bit performance from your 386. Our NDP C compiler takes the original C concept of writing lower level code with a higher level language to the limit by providing an inline assembly language interface that lets the programmer specify which register is used to store a variable. This feature makes it possible to use ports or perform interrupts or block moves inline, instead of through calls. The use of register aliased variables to control hardware reduces the size of critical code sequences by a factor of 3 to 10 and keeps the 386's pipelines full by eliminating costly calls. If you are interfacing DOS or the ROM BIOS, or writing graphics routines, a device driver, operating system kernel, or an embedded application, you owe it to yourself to try NDP C-386.

Our Users Report:

Milt Capsimolis of Ithaca Software in Ithaca, NY, developer of HOOPS, the highly regarded 3D, object-oriented graphics library, reports, "We ported a huge library — well over 100,000 lines — without a hitch, in less than a day! NDP C-386 was the fastest 386 compiler we used...We also liked the enormous advantage it offers through its support of the Weitek coprocessor."

Fred Ziegler of AspenTech in Cambridge, Mass. reports, "I ported 900,000 lines of FORTRAN source in two weeks without a single problem!" AspenTech's Chemical Modeling System is in use on mainframes worldwide and is probably the largest application to ever run on an Intel processor.

Please call (508) 746-7341 for more information.

NDP C is also the language of choice if you are combining C with FORTRAN or Pascal, are planning to use any one of four coprocessors that run with the 80386, or require the highest globally optimized code attainable. MicroWay's C, FORTRAN and Pascal compilers come with a 70 function, device-independent graphics library that automatically supports Monochrome, Hercules, CGA, EGA, and VGA adapters and makes it easy to interface memory-mapped peripherals such as digitizers or serial devices such as mice. We make it possible for you to write your own numeric exception handler and include examples written in C, FORTRAN and Pascal. Finally, our C is not only one of the easiest to use, but supports two dialects: ANSI and UNIX. In fact, our UNIX implementation is so close to the standard, that our best customer has become AT&T!

386 Compilers and Tools

NDP Fortran-386™, **NDP C-386™**, and **NDP Pascal-386™** compilers generate globally optimized, mainframe quality code that runs on the 386 or 486 in protected mode under UNIX, XENIX or Phar Lap extended DOS. The compilers address 4 gigabytes of memory while supporting the 80287, 80387 and Weitek coprocessors. They all come with a library of over 70 device-independent graphics, keyboard and sound routines. Applications can mix code from all three compilers and assembly language. The DOS versions allow the user to write his own numeric error handlers and interface 386 real mode programs from protected mode. The VM versions use Phar Lap's Virtual Memory Manager to run programs which exceed the size of your system memory. **NDP Fortran-386** is a full FORTRAN 77 with FORTRAN 66, BSD 4.2, DOD and VMS extensions. **NDP C-386** is a full K&R C with both MS and ANSI extensions. It is 100% compatible with UNIX C and is substantially faster than the C which comes with UNIX. **NDP Pascal-386** is a full ANSI/IEEE Pascal, with extensions from C and BSD 4.2 Pascal.

DOS versions (require Phar Lap Tools) . $595
VM version (requires VMM) $695
UNIX/XENIX versions $795
Phar Lap Development Tools $495
Phar Lap Memory Manager (VMM) . . .$295

NDPWindows™ — 80 functions which create, store, and recall menus and windows. Works with NDP C and drives all popular graphics adapters . . . . . Library: $125, C Source: $250

NDP HOOPS™ — An NDP port of Ithaca Software's HOOPS, this 3D object-oriented graphics library, callable from NDP C, makes it possible to develop full-featured 386 CAD packages and front ends $575

NDP Plot™ — Calcomp compatible plot package, callable from NDP Fortran. Includes drivers for popular plotters and printers. Works with MDA, CGA, EGA and VGA $325

NDP/FFT™ — The fastest running FFTs on a PC! 40 hand-coded routines that handle 1 and 2 dimensional data arrays. Includes an in core solver that spills to disk for arrays too large to fit in memory.

8088/87/286/287/386/387 version $250
Protected mode NDP version $250

NEW! C++

NDP C++ is a MicroWay port of the UNIX C++ preprocessor version 1.2. It runs in protected mode on DOS, UNIX or XENIX, and is ideal for writing numerics and graphics applications. The product comes with an example of how to support complex numbers in C++ $495

Parallel Processing

Videoputer™ — The highest performance graphics card on the market. Uses a T800 and TI 34010 in conjunction with an 80 MHz Brook-Tree DAC. . . . . With one meg $4495

Monoputer™— Includes one T800 and up to 16 meg of RAM for parallel code development. The four MWhetstones T800 makes this the ideal FORTRAN engine for cost-effective execution of your mainframe programs. $1295

Quadputer™— This board for the AT or 386 can be purchased with 1 to 4 transputers and 1 or 4 meg of memory per transputer. Two or more Quadputers can be linked together to build networks of up to 100 or more transputers providing mainframe power . . . . . from $1995

Transputer Compilers and Applications
These parallel languages are designed for use with either a Monoputer or Quadputer.

Logical Systems Parallel C $595
3L Parallel C, FORTRAN, or Pascal . $895
ParaSoft EXPRESS Package:— Includes transputer communications libraries, C source level debugger and system Performance Monitor. $1500
COSMOS/M — Finite element analysis. Comes bundled with a Quadputer. Performs at near supercomputer speeds . . . . . from $5995
T800/NAG™ — Features a library of 268 engineering and scientific numerical algorithms. Callable from 3L Fortran. $2750

NDP to HALO '88 Graphics Interface — Lets you call HALO '88 from NDP compilers . . . $100

387BASIC™ — Our 16-bit MS compatible compiler introduces numeric register variables to produce the fastest running 80x87 code on the market $99

386 Your AT

386/387Turbo-AT™ — This board plugs into the 80286 socket, allowing your IBM AT to run 32-bit protected mode code written for the 80386. Includes an 80387 socket. The most cost-effective AT upgrade!

386/387Turbo- AT 16MHz $495
386/387Turbo-AT 25 MHz $695

Weitek-Based Coprocessor Boards

mW1167™ and **mW3167™** coprocessor boards are built at MicroWay using Weitek components. Each includes an 80387 socket.

mW1167-16 . . . . . $895
mW1167-20 . . . . . $1095
mW1167 Microchannel-16/20 . . from $995
mW3167 Microchannel-25/33 . . . from $1795
3167-20 . . . . . $995
3167-25 . . . . . $1295
3167-33 . . . . . $1695
mW3167/80387 Board $150

Intel Coprocessors and RAM

All coprocessors include our 87Test program.

8087	$84	**8087-2**	$120
80287-8	$195	**80287-10**	$220
80387-16	$350	**80387-16SX**	$310
80387-20	$400	**80387-25**	$500

80C287A (CMOS) $280
287Turbo-12 (for AT compatibles) . . . $350
RAMpak™ - one meg 32-bit memory module for Compaq 386 20/25 . . . . . $395
RAMpak™ - four meg module $1495
Please call for our 100 and 80 ns RAM prices.

Multi-User Accelerators

MicroWay's **AT8™** and **AT16™** intelligent serial controllers run 8 to 16 terminals under UNIX or XENIX without bogging down your AT, 80386 or PS/2. **AT8**: $895 **AT16**: $1295

12 MHz PC Accelerators

SuperCACHE-286 12 MHz $399
FastCACHE-286 12 MHz $299
FastCACHE-286 9 MHz $199

NUMERICS HOTLINE
(508) 746-7341

MicroWay

World Leader in PC Numerics

Corporate Headquarters: P.O. Box 79, Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466
USA FAX 508-746-4678 Italy 02-74.90.749 Holland 40 836455 Germany 069-75-2023

Number **104** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Four Gigabytes in Real Mode

by Thomas Roden

Until now, all *X*86 family processors have been limited to 1M (or 1M + 64K - 16 in the case of the 80286) of immediately available linear memory when running under real-address mode (real mode). With the advent of the 80386 and 80486 processors, this is no longer the case. With a little manipulation at initialization time, it is possible to address all 4 gigabytes of physical address space in real mode without the overhead of virtual 86 mode. In addition, it is not necessary to violate DOS to do so.

While real mode of the 80386 family of processors is generally viewed as a mode that is equivalent to the normal operation of the 8086 and 8088 processors, this is not entirely true. The real modes of these processors are intended to be only compatible with—not identical to—lesser processors of the *X*86 family. This is borne out by the fact that extended registers and the additional segment registers FS and GS are available in real mode.

It is not illegal to use the extended registers for addressing with the additional segment registers, as in the following statement:

```
MOV   DWORD PTR GS:[EAX], EBX
```

The only proviso is that EAX must be less than 64K; otherwise, a general protection fault results. It seems curious that such an error can occur in real mode—a mode where protection is nominally unused.

Another curious feature is that when returning to real mode from

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

protected mode by clearing the Protection Enabled (PE) bit in Control Register 0 (CR0), the system is required to first load all segment registers with selectors that reference segments of size 64K (see the Intel *80386 Programmers Reference Manual*, Ahearn-Wallstrom, 1986). When this is not done, general protection faults can occur upon access of those segments.

This behavior suggests that the limit-checking aspects of protection are active in real mode, but the default state (set at power on or reset) is such that the limits of the segments are all set to 64K - 1, normally obscuring the fact that limit checking is truly in force.

A way to test this is to have a program enter protected mode, set a segment register to refer to a selector of size greater than 64K, return to real mode, and attempt to access data farther than 64K from the beginning to the segment in question.

Tests on the 80386DX, 80386SX, and 80486 have shown that it is indeed possible to reference data more distant than 1M + 128K (120000h) from the beginning of a segment, and there is no reason to believe that this behavior is not consistent for the entire 4 gigabytes that can be referenced by an extended register.

In general, the 80386 and 80486 can be used as more than a fast 8088 without having to support an operating system other than DOS.

This technique has many potential uses. Applications that are 386 aware can run in the faster real mode and still access all linear memory available in the system without disturbing DOS. System software can access normally hidden memory-mapped I/O locations without the overhead it takes to change processor modes. In general, the 80386 and 80486 can be used as more than a fast 8088 without having to support an operating system other than DOS.

A danger of using this technique is that software that performs a hardware reset (normally via the keyboard controller to return from protected mode to real mode) will cause the segment limits to return to 64K - 1. This can be handled by an Interrupt Service Routine (ISR) that detects when general protection faults are caused by an attempt to address in this fashion and reinstates the extended limits.

It's important to remember that this is a system-oriented technique and that it risks collision with other techniques that attempt to allow greater memory access. Most notably, if this is attempted from a virtual 86 task, it will definitely cause a general protection fault. More generally, any extended memory manager will be in danger of colliding with data accessed via the extended registers.

Only data-segment limits have been explored with this technique; however, it may be possible to perform the same sort of manipulations on the Default Bit (D-bit) of code segments to have full use of the 32-bit Extended Instruction Pointer (EIP) as well. Similarly, it may be possible to affect the Expand-Down (E-bit) to instill better stack-underflow recovery in real mode as well.

It is important to keep in mind that the 80386 family of processors is extremely powerful. Even in real mode, these processors allow new features that have yet to be fully investigated and exploited. PJ

Thomas Roden is a Software Engineer for AST Research, Inc., of Irvine, California, where he is working on software validation of system integrity and general pontification. Readers may write to Thomas in care of AST Research, Inc., 2121 Alton Ave., Irvine, CA 92714.

Code follows.

C Programmers
Explore a world of DOS memory
Beyond 640K

Use the PowerSTOR C library to exploit:
- LIM 4.0 or LIM 3.2Expanded Memory
- AT Style Extended Memory
- Hard Disk Space
- Any Combination

- PowerSTOR is the easy way to use system memory resources. It self-configures at run time, under control of your software.
- PowerSTOR makes system memory resources available to your software as STORheaps. STORheaps can be allocated and de-allocated dynamically. Up to a gigabyte of PowerSTOR memory can be used as one huge STORheap or many smaller STORheaps.
- PowerSTOR uses none of the normal heap and it minimizes its usage of stack space. These valuable resources are preserved for your use.

Library with K & R standard source code — only $92.50
(includes shipping)

Order today and explore beyond the limits.

Acme Software

P. O. Box 40734, Portland, Oregon 97240 — Phone (503) 235-2816

Number 305 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ CODE
All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 96.

Code from "Four Gigabytes . . . ," by Thomas Roden

1

```
LISTING 1
*****************************************************************
;*
;*      TOMCOM.ASM -    .COM program to relax segment limit on GS in
;*                      real-address mode.
;*
;*      Assembled using Microsoft Macro Assembler 5.1
;*
;*****************************************************************

;*      Need to use 80386 protected mode instructions
        .386P

;
;       JUMPFAR does a far jump in USE16 segments
;
JUMPFAR MACRO   PARM1,PARM2
        DB      0EAH                    ; jump far direct
        DW      (OFFSET PARM1)          ; to this offset
        DW      PARM2                   ; in this segment
        ENDM

CMOS_ADDR               EQU     0070H
CMOS_DATA               EQU     0071H

SYS_PROT_CS             EQU     0008H
SYS_REAL_SEG            EQU     0010H
SYS_MONDO_SEG           EQU     0018H

cr              equ     0dh
lf              equ     0ah
dos_call        equ     21h
print_func      equ     09h

cseg    segment use16 public    'code'
        assume  cs:cseg,ds:cseg,es:cseg,ss:cseg

        org     100h
begin:
;
;       Setup Environment
;
                mov     sp,offset stk_top               ; setup stack
                mov     ax,cs                           ; store program segment

;       WARNING:  Self-modifying code.  Not needed if done in ROM.
                mov     word ptr cs:self_mod_cs,ax

                mov     ds,ax                           ; set ds := cs
                jmp     init_com                        ; go do the work

;-----------------------------------------------------------------
;
;       Variables
;
;-----------------------------------------------------------------
com_gdt_ptr     dq      ?                               ; for LGDT
;               Limit   Base    Rights  LimitHi
com_gdt         dw      00000h, 00000h, 00000h, 00000h  ; unusable
                dw      0FFFFh, 00000h, 09A00h, 00000h  ; code seg
                dw      0FFFFh, 00000h, 09200h, 00000h  ; data seg
                dw      0FFFFh, 00000h, 09200h, 0008Fh  ; mondo seg (4Gb)
com_gdt_end     label   word
;-----------------------------------------------------------------

load_msg        db      'Segment Helper Loaded',cr,lf
                db      'Use extended register and GS: to activate',cr,lf,'$'

fixup_Msg       db      'System Altered',cr,lf,'$'

                db      255 dup(0ffh)                   ;local stack
```

2

```
stk_top         db              (0ffh)


;*****************************************************************
;*      Go to protected mode to relax limit on GS                *
;*      Entry   -       none
;*      Exit    -       gs: invisible selector information fixed up
;*                      ds & ss: set to cs (ok for COM file)
;*      WARNING:        MUST EXECUTE BELOW 1M, GATE A20 NOT DISABLED
;*****************************************************************
kill_seg_limit  proc    near

                mov     ax, cs                          ; get linear address
                movzx   eax, ax
                shl     eax,4

                mov     ebx, eax                        ; store copy of CS linear
                mov     word ptr cs:com_gdt+10, ax      ; store in code segment desc
                mov     word ptr cs:com_gdt+18, ax      ; store in data segment desc
                ror     eax, 16                         ; swap words
                mov     byte ptr cs:com_gdt+12, al      ; bits 16-23
                mov     byte ptr cs:com_gdt+20, al      ; bits 16-23
;
;       Setup Limit and Base for GDTR
;
                add     ebx, offset com_gdt
                mov     word ptr cs:com_gdt_ptr, (offset com_gdt_end - com_gdt - 1)
                mov     dword ptr cs:com_gdt_ptr+2, ebx

;       store flags for restoring after cli
                pushf
;       do NOT allow interrupts as IDT is not valid in protected mode
                cli
;       disable NMI's here
                in      al,CMOS_ADDR
                mov     ah,al
                or      al, 080H                ; don't disturb rest of 70
                out     CMOS_ADDR, al
                and     ah, 080H
                mov     ch, ah                  ; store old state of NMI mask

                lgdt    fword ptr cs:com_gdt_ptr        ; assume DOS isn't using this

                mov     bx, cs                  ; save com segment

                mov     eax,cr0
                or      al,01                   ; set PE bit
                mov     cr0,eax                 ; protection enabled
                jumpfar ksl_pmode,SYS_PROT_CS   ; purge queue and fix CS

ksl_pmode:
                mov     ax, SYS_REAL_SEG        ; prepare limits on segments
                mov     ss, ax
                mov     ds, ax
                mov     es, ax
                mov     fs, ax
;
;       Here are the instructions that makes it all possible
;
                mov     ax, SYS_MONDO_SEG               ; gs will now be 4G
                mov     gs, ax

                mov     eax,cr0
                and     al,NOT 01               ; clear PE bit
                mov     cr0,eax                 ; protection disabled
;
;       the following relies on self-modification performed at beginning
;       of program.  The following macro would be useful here for ROM code only,
;       so it is manually expanded to facilitate the (yech) self-modification.
;
                jumpfar ksl_rmode,SYS_REAL_CSEG ; purge queue and fix CS
;
                DB      0EAH                    ; jump far direct
                DW      (OFFSET ksl_rmode)      ; to this offset
;       (please don't send me to programmer's hell for this one)
self_mod_cs     DW      ?                       ; in this segment
ksl_rmode:
```

Code continues . . .

PRODUCED 2014 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

```
                mov     ss, bx                  ; get seg regs back
                mov     ds, bx
                xor     ax, ax                  ; clear unused to be cleanly
                mov     es, ax
                mov     fs, ax
                mov     gs, ax
;
;       back in Real Mode, IDT is ok again
;
                in      al,CMOS_ADDR
                and     al, 07FH
                or      al, ch                  ; restore old state of NMI mask
                out     CMOS_ADDR, al
;
;       note that 386 does not have POPF problem
;
                popf                            ; restore IRQ mask

                ret

kill_seg_limit  endp


;*************************************************************
;*      Main Control Routine Starts Here                     *
;*************************************************************
initialize      proc    near
        assume  ds:cseg
;
init_com:

;display message and then relax limit checking on GS
;
        mov     dx,offset load_msg
        mov     ah,print_func
        int     dos_call
;
; fix gs
;
        call    kill_seg_limit

        mov     dx,offset fixup_Msg
        mov     ah,print_func
        int     dos_call
;
;check amount of memory to retain
;
        mov     ax,4C00h        ;Terminate with return code function
        int     dos_call
;
initialize      endp
;-------------------------------------------------------------


cseg    ends
        end     begin
```

LISTING 2

```
/*
 * tompeek.c
 *
 * High memory peek utility
 *
 * This program returns a screen dump of ram (by byte) at the provided
 * linear (which is also physical) address for the provided number of bytes.
 *
 *              Compiled using Microsoft C 5.1--Small Model
 *
 */
#include <stdio.h>
#include <dos.h>

/* Main function.
 *
 * Check command line information, and perform the appropriate hex-dump
 *
 */
main(argc, argv)
int     argc;
char    **argv;
{
        short   size;
        long    source;
        void    tomPeek(long,short);

        if(argc==3)
                {
                sscanf(argv[1],"%lx",&source);
                sscanf(argv[2],"%x",&size);
                tomPeek(source,size);
                }
        else
                {
                printf("usage: TomPeek <address> <size>\n");
                }

}

/*
 *      tomPeek -       Dump the memory at linear address "source" for "size" bytes
 *                                      of sixteen-byte lines
 */
void    tomPeek(source,size)
long    source;
short   size;
{
        char    peekBuf[16];
        short   i,j,k;
        extern short near       get_high_mem(char *,short,long);

        for(i=0;i<size;i+=16)
                {
                j = size-i;
                if(j>16)
                        j = 16;
                get_high_mem(peekBuf,j,source);
                source += j;
                for(k=0;k<j;k++)
                        {
                        printf("%02X ",((short)(peekBuf[k]))&0xFF);
                        }
                printf("\n");
                }
}
```

4

LISTING 3

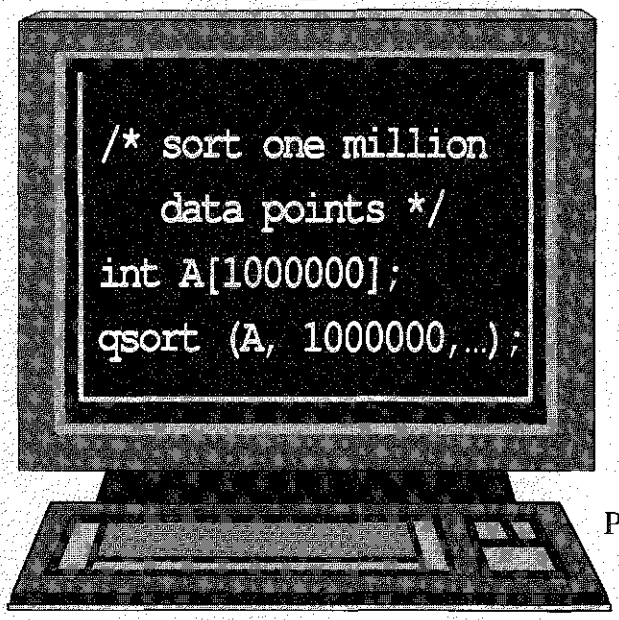
```
;*************************************************************
;*
;*      PEEPER.ASM -    Assembly subroutines to move memory from a 4G linear
;*                      address into a buffer in the current data segment
;*
;*      Assembled using Microsoft Macro Assembler 5.1
;*
;*************************************************************

;*      Need to use 80386 protected mode instructions
        .386P


KBC_IBF_BIT             EQU     02H
KBC_WOP_CMD             EQU     0D1H
KBC_ROP_CMD             EQU     0D0H
A20_OFF_DATA            EQU     0DDH
A20_ON_DATA             EQU     0DFH
KBC_NOP_CMD             EQU     0FFH

KBC_DATA_PORT           EQU     060H
KBC_CMD_PORT            EQU     064H
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MS-DOS 386•486

Can your C compiler handle this?



Version 1.6

MetaWare's High C® and Professional Pascal™ —

the *first* C and Pascal compilers to generate protected-mode 386 code for any 386 MS-DOS machine.

And the compilers themselves can run in protected mode, giving you the power to create large programs.

MetaWare's compilers and Phar Lap's 386|DOS-Extender let you expand your application to the 32-bit registers and address space of the Intel 386 and 486.

	MetaWare High C	Microsoft C	Borland Turbo C
Generates 32-bit native-mode code	✔	-	-
Arrays up to 4 gigabytes	✔	-	-
Demand-paged virtual memory	✔	-	-
387 in-line transcendentals	✔	-	-
Weitek floating-point support	✔	-	-
Dhrystones - 386	12,853	not available	not available
- 286	9,956	7,142	8,333

Practically every 386 protected-mode MS-DOS program available is done with MetaWare's High C or Professional Pascal.

Industry leaders such as

- ***Autodesk - AutoCAD***
- ***Borland - Paradox 386***
- ***Fox Software - FoxBase+/386***
- ***IBM - Interleaf Publisher***
- ***Wolfram Research - Mathematica***

and more use MetaWare's compilers to get dramatic increases in performance. Don't let *your* 386 power go to waste!

Call now for your 386 compilers and DOS extenders: (408) 429-6382.

MetaWare®
INCORPORATED

Compiler Products for Professional Software Developers

2161 Delaware Avenue • Santa Cruz, CA 95060-5706 • Phone (408) 429-META(6382) • FAX (408) 429-WARE(9273)

MetaWare and High C are registered trademarks and Professional Pascal is a trademark of MetaWare Incorporated. All other products are trademarks of their respective companies.

Number **45** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

5

```
GA20_FAIL_ERR_CODE      EQU     0FFFFH

_TEXT   segment use16 public    'CODE'
        assume  cs:_TEXT,ds:_TEXT,es:_TEXT,ss:_TEXT


;*****************************************************************
;*      get_high_mem(dest,len,source) - copy memory from source to dest (len bytes)
;*      Entry   -       dest -          buffer in current DS
;*                      len -           size of buffer in bytes
;*                      source -        32-bit linear address of source memory
;*      Exit    -       gs: set to zero
;*      WARNING:        Forces GateA20 low when done
;*****************************************************************
                public  _get_high_mem
ghm_dest        equ     word ptr [bp+4]
ghm_len         equ     word ptr [bp+6]
ghm_source      equ     dword ptr [bp+8]
_get_high_mem   proc    near
                push    bp
                mov     bp, sp
                push    di

                mov     al, 080h                        ; allow normal A20
                call    gate_A20

                xor     ax, ax
                mov     gs, ax                          ; zero out gs
                mov     ax, ds
                mov     es, ax                          ; es := ds
                mov     di, ghm_dest
                mov     cx, ghm_len
                mov     ebx, ghm_source
                jcxz    ghm_bottom
ghm_top:
                mov     al, byte ptr gs:[ebx]
                inc     ebx
                stosb
                loop    ghm_top
ghm_bottom:
                xor     al, al                  ; cripple A20 for DOS
                call    gate_A20

                pop     di
                pop     bp
                ret

_get_high_mem   endp

;*****************************************************************
;*      gate_A20(how)   -       set gateA20 to the desired state
;*      Entry   - AL==80h => A20 on
;*                AL==00h => A20 off
;*      Exit    -       GateA20 in desired state
;*      WARNING:        Assumes preferred states of output port bits
;*****************************************************************

;
;       gate_A20 -      set gateA20 to the desired state
;               Entry - AL==80h => A20 on (no 1M wrap)
;                       AL==00h => A20 off (1M wrap)
;
gate_A20        proc    near
                test    al,080H                 ; al&80 = 0 ?
                jnz     short ga20_on
;       al is zero, turn a20 off
                mov     ah,A20_OFF_DATA         ; kbc off gate a20
                call    empty_KBC               ; clear to use
                jnz     short ga20_err          ; didn't clear
                jmp     short ga20_KBC_ctl      ; cleared, go fix KBC A20
ga20_on:
                mov     ah,A20_ON_DATA          ; kbc on gate a20 if needed
ga20_KBC_ctl:
                mov     al,KBC_WOP_CMD          ; Write Output Port
                out     KBC_CMD_PORT,al         ; command it
                call    empty_KBC
                jnz     short ga20_err
                mov     al,ah
```

6

```
                out     KBC_DATA_PORT,al
                call    empty_KBC               ; wait till it clears
                jnz     short ga20_err          ; didn't clear
                mov     al,KBC_NOP_CMD          ; flush it through
                out     KBC_CMD_PORT,al         ; command it
                call    empty_KBC
                jnz     short ga20_err
;       we're here, it worked!!
ga20_ok:
                xor     ax,ax
                jmp     short ga20_out
ga20_err:
                mov     ax,GA20_FAIL_ERR_CODE
ga20_out:
                ret
gate_A20        endp

;
;       empty_KBC -     Empty keyboard controller input buffer
;               Entry - none
;               Exit  - bashes AL
;
empty_KBC       proc    near
                push    cx                      ; save reg
                xor     cx,cx                   ; set for max timeout
empty_KBC_loop:
                in      al,KBC_CMD_PORT         ; get KBC status
                test    al,KBC_IBF_BIT          ; check IBF bit
                loopnz  empty_KBC_loop          ; try till timeout
                pop     cx                      ; restore reg
                ret                             ; that's all folks
empty_KBC       endp



_TEXT   ends
        end;
```

LISTING 4

```
masm /Zi tomcom;
link tomcom /co;
exe2bin tomcom.exe tomcom.com
del tomcom.exe
```

LISTING 5

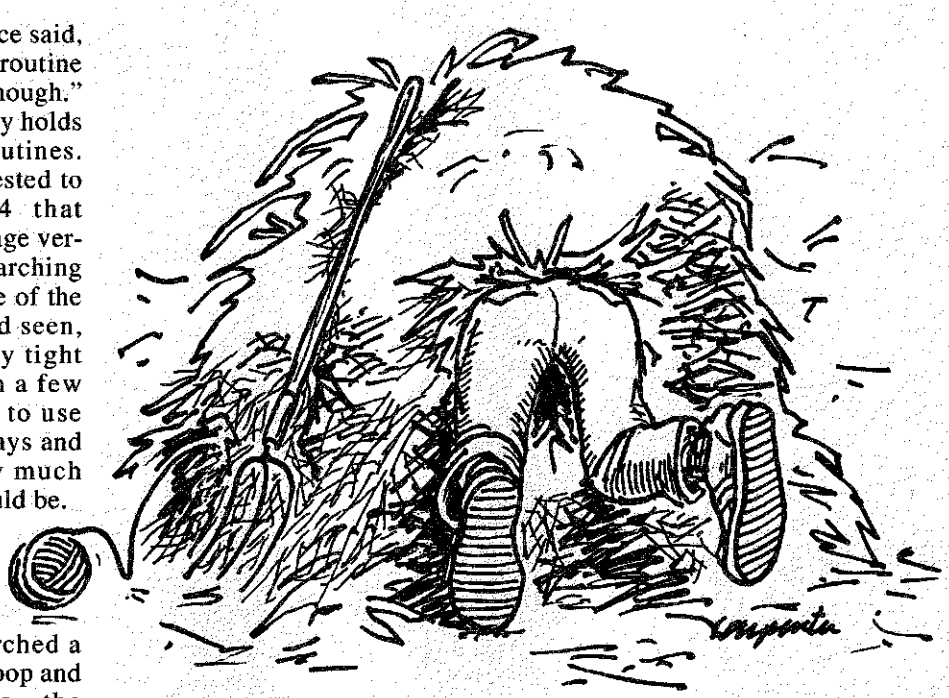
```
TOMPEEK.OBJ : TOMPEEK.C
    CL /c /Zi TOMPEEK.C

PEEPER.OBJ : PEEPER.ASM
    MASM PEEPER;

TOMPEEK.EXE : TOMPEEK.OBJ PEEPER.OBJ
    LINK /CO TOMPEEK+PEEPER;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

QuickBASIC's Fast String-Searching Algorithm

by Ethan Winer

A wise programmer once said, "You can never write a sort routine that's fast enough or small enough." No doubt this same philosophy holds true for string-searching routines. Therefore, I was quite interested to see an article in *PJ* 7.4 that presented an assembly language version of the Boyer–Moore searching algorithm. Compared to some of the other implementations I had seen, this one appeared to be very tight and well coded. I'd written a few searching routines myself to use with QuickBASIC string arrays and was very eager to see how much faster this "new" method would be.



After carefully modifying the printed routine to work with QuickBASIC 4.5, I devised a short test program to time it. My benchmark searched a 10,000-character string in a loop and compared two algorithms—the Boyer–Moore routine published in *PJ* and QuickBASIC's built-in INSTR function. However, based on the results I obtained, it appears that the Boyer–Moore method is little more than a mathematical curiosity for my test cases. Indeed, it was only half as fast as QuickBASIC!

How QuickBASIC Does It

So how does QuickBASIC search strings so fast? A quick examination under Microsoft CodeView reveals a fundamentally simple, yet elegant, algorithm. I have adapted QuickBASIC's method (with permission from Microsoft) and named it SearchString. This is shown in Listing 1.

(Please understand that this routine is not intended for use with QuickBASIC; rather, it provides a foundation for achieving BASIC's high performance in other high-level languages. If you do intend to try SearchString with QuickBASIC, bear in mind that it has been designed as a function and therefore must be declared. Assembly language functions were added to QuickBASIC beginning with version 4.00, and they are one of the most important features that Microsoft has added to the language.)

Three passed parameters are required to specify the starting character, the string being searched, and the substring to locate. Even though this version uses the same basic al-

Continued on page 97

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

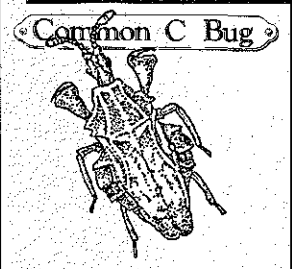
T-SHIRTS

Multi-color imprints on the highest quality 50/50 cotton T-shirts.

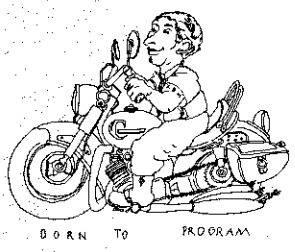
SIZES S, M, L, XL

NOW ONLY $10.00 each*
(includes shipping & handling)

PJ ISSUE SPECIAL!
$2.00 OFF REGULAR PRICE*



COMMON C BUG



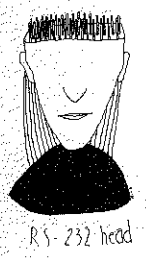
BORN TO PROGRAM

T-SHIRTS
GET THEM NOW FOR LESS!

Stock up now on all the great *PJ* T-shirts you've wanted for so long for just $10.00 each. Cover your torso with these unique expressions of your chosen profession. They also make great Christmas gifts, so order a bunch today.

"(Offer expires December 31, 1989. All orders must be prepaid.)

PARALLEL PORTS



RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

BACK ISSUES

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 6.1: Consulting is an attitude; write mode 3 of the VGA; the LIM 4.0 expanded memory specification; environment variables in Turbo Pascal 4.0; a timer program in C; porting mainframe software to the PC—FORTRAN compilers under DOS and the Definicon 32-bit coprocessing engines.

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink Noise; yet another VGA write mode; inside Turbo Pascal 4.0; the proposed ANSI C language standard; converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; *The Norton Guide for BIOS/DOS Assembly*.

Vol. 6.3: Hal Hardenbergh on memory trends; Pink Noise; getting your consulting tools in order; Gordon Letwin on OS/2 memory management; virtualizing the 386; Pascal—a CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multiline functions.

Vol. 6.4: Job shopping; windowing systems overview; reading VGA memory; X Window—a view from DOS; a digital clock in C for MS Windows; windowing in TopView, DESQview, and MS Windows; MS C 5.1 and the ANSI standard; a mouse unit for Turbo Pascal 4.0.

Vol. 6.5: Hal Hardenbergh on computers in the fast lane; Consulting—no-win jobs and too much work; VGA text paging; using dynamic arrays and records in Turbo Pascal 4.0; an OS/2 Presentation Manager application template; WATCOM C 6.0 and the ANSI standard; object-oriented OS/2 in C++; Ada and software engineering; REXX on the PC.

Vol. 6.6: PC DOS 4.0; OS/2 vs. UNIX; VGA screen saving and color selection; MASM quirks; addressing extra memory for DOS; DOS device drivers debugging and information utility; proposed ANSI C language standard; Ray Duncan on Forth; Ada data types and exceptions; PC floating-point hardware and software.

Vol. 7.1: GRAPHICS: High-resolution VGA color modes; graphics processors; modifying type constants stored in Turbo Pascal 4.0 and 5.0 compiled EXE code; comparing Pascal overlay managers; JPI's TopSpeed Modula-2; faster QuickBASIC code; Microsoft's CodeView debugger; Lattice C V3.3; smart terminal emulations in Windows; the COBOL-85 ANSI standard; Pink Noise—industry news and reviews.

Vol. 7.2: OPERATING SYSTEMS: undocumented DOS features; fast 80386 block moves under DOS; OS/2 dynamic link libraries; OS/2 multitasking; debugging utility for PM and Windows; EGA and VGA animation; disk formatting in Pascal; how to become a reseller; reviews of windowing and graphics tools, disassemblers, and *The 80386 Book*.

Vol. 7.3: DATABASE JUNGLE: neural nets; Michael Abrash's "On Graphics"; embedded SQL for a relational database; Btrieve; converting database files using filters; the proposed ANSI C language standard; a substitute for the missing TYPE command in Windows; user interface tools for DOS; reviews of dBASE C libraries, software tools, FORTRAN tools, conferences, exhibits, and books.

Vol. 7.4: COMMUNICATIONS "Pink Noise," by Frank Greco; Michael Abrash's "On Graphics"; a simple terminal program in PM; communications and protocols; better coding conventions for assembly language; implementing the Boyer–Moore search algorithm in assembly language; database normalization theory and practice; Turbo Pascal 5.5 development tools; Lint utilities for the PC; a millisecond timer for BASIC; exit codes for QuickBASIC; reviews of RamNet, DOS/16M, PCX *Technical Reference Manual* and Programmer's Toolkit, GRASP 3.1, AXOS, and *Microsoft Mouse Programmer's Reference*.

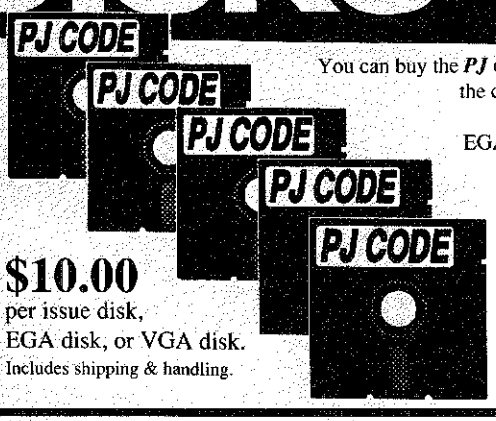
Vol. 7.5: OBJECT-ORIENTED PROGRAMMING Class libraries in C++; from C to objects; improved C++ code generation; Hal Hardenbergh on the RISC/CISC schism; handling version management on large projects; "Still-Higher 256-Color VGA Resolutions," by Michael Abrash; a library system for assembly language; adding world scaling to Turbo Pascal's BGI graphics primitives; numeric extensions to C; reviews of 386 development environments and tools, mini-case tools, and Robert Sedgewick's book *Algorithms*.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY.

CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



You can buy the *PJ* code listings from volume 5.5 to the current issue. Code disks for the Michael Abrash "On Graphics" EGA/VGA series are also available.

Diskettes are available for the following issues:

5.5	6.3	7.1	7.5
5.6	6.4	7.2	7.6
6.1	6.5	7.3	EGA
6.2	6.6	7.4	VGA

360K MS-DOS disks standard; other formats available.

$10.00 per issue disk, EGA disk, or VGA disk.
Includes shipping & handling.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

gorithm as QuickBASIC, it is slightly faster, because it does not check for an illegal Start parameter. Also, the start variable is passed "by value," rather than by address, which is the method used by most high-level languages. This eliminates an extra step of locating the variable within the assembly language routine, thus saving a few bytes and clock cycles.

Rather than relying on a purely "brute-force" approach that compares every character in both strings, QuickBASIC instead isolates the first character. In this case, it is loaded into AL. Then the 8088's speedy **scasb** instruction is used to find the first occurrence where a possible match might begin. The beauty of this approach is twofold: **scasb** is the fastest way to find a single matching character, and subsequent searches require comparing one fewer character in each string. SearchString begins by obtaining the starting character offset in the string being searched. Even though most situations would require searching the entire string, being able to resume a search in the middle of the string is often valuable. Next, SI and DI are loaded with the address of the first character in each string, and CX is assigned the number of characters to search. Unlike C or Pascal, BASIC maintains a string descriptor that tells how long a string is and where in near memory its data is located. The final initialization steps are to adjust DI to point to the correct portion of the string being searched and to load AL with the first character of the string to find. The main loop begins at the *Scan* label and ends at the label *Found*. When **scasb** finds a character that matches the one in AL, the remaining characters in both strings are compared. If a match is found, SearchString returns to the caller with AX holding the offset into the source string. Otherwise, **scasb** is used again to find the next occurrence of the first character. This is repeated until either the search string is located or the source string is exhausted.

Rather than relying on a purely "brute-force" approach that compares every character in both strings, QuickBASIC instead isolates the first character.

Adapting the Routine

You can easily modify SearchString to work with C, Pascal, assembly language, or whatever by simply changing how the various registers are initially loaded. For example, the length of a string in Pascal is stored in the byte that precedes its first character. Conversely, C uses a zero byte to mark the end of its strings, which means that extra steps (and time) will be needed to determine their length before beginning the search. SearchString could also be

Prolex

The Programming Language You've Always Wanted

As computer consultants, we are faced daily with the sorts of problems that no one tool can solve simply. For instance, the problem of converting huge piles of C source to use different library routines for I/O, or extracting and parsing SQL 'select' statements to determine which tables they select. For some, AWK and its variants may be made to work. For most a REAL program in a REAL programming language is needed.

Of course, we also have to write BIG programs—like intelligent code generators, expert systems for diagnosing faults in speech synthesis systems, and tools for system configuration and installation.

NO PROGRAM TOO SMALL — OR LARGE

That's why we invented PROLEX. PROLEX is an interpreted language that allows us to solve everyday problems in minutes, yet gives us the power and flexibility to solve the big problems as well.

THE LANGUAGE YOU REALLY WANTED

We devised PROLEX for ourselves, but we will send YOU a copy for only $99.95. Even if you just use PROLEX to help you create new programming tools, it will be an excellent investment. In fact, you probably spend more on toner or ribbons for your printer every year.

AND IT DOESN'T COST MUCH

After you've used it, you won't be able to IMAGINE doing without it.

FEATURES

■ Boundless Arrays and Tables	No array bounds. No bounds on the size of tables. All structures creatable and modifiable at run-time.
■ No TYPE restrictions	Tables and arrays may contain elements of any type and size.
■ Arbitrary-Precision Arithmetic	Floating point and integer numbers may be of any size.
■ Powerful SNOBOL-like string and pattern-matching facilities	Parse and reconstruct strings in the most complex of formats effortlessly.
■ Forward and Backward chaining inference rules	Rule-based programming features for expert systems development BUILT INTO the language.
■ Integrated Symbolic Debugger	Available at all times. May be entered with a keystroke. Makes understanding and debugging programs trivial.
■ Extensible	Create your own language features and operations using C.
■ Object Oriented Programming Facilities	Supports Objects and Inheritance Multiple Class Levels

ORDER YOUR COPY TODAY.
We will ship C.O.D. Free. All you pay is $99.95. Checks accepted. XENIX 286/386 Available. **Call (615) 297-2298**

METALEX systems, inc.
1904 BELCOURT AVENUE
NASHVILLE, TENNESSEE 37212

Number **325** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Six easy ways to boost your BASIC

ProBas™ Basic Programming Library

So who cares that BYTE magazine calls **ProBas** a "Supercharger for QuickBASIC" or that PC Tech Journal says that **ProBas** is a "high-quality, high-quantity package"? Who buys a product just because Jerry Pournelle said "Anyone doing serious QuickBASIC programming would do well to get [**ProBas**]"? And who cares that Wayne Hammerly calls **ProBas** "The greatest thing since sliced bread"?

Who?--Only those who want to write better, faster, slicker programs and save hundreds of programming hours in the process. With all of that hoopla out of the way, we are formally announcing the momentous release of **ProBas** Version 3.1, now with over 400 assembly routines to make BASIC programs faster and more powerful than you ever dreamed with features like:

- A 1,000-page two-volume manual
- Full mouse support
- Extended and EMS memory support
- Full-featured windowing
- Moveable, resizable windows
- Screen snapshots (text & graphics)
- Virtual screens in memory
- Lightning-fast file I/O
- Critical error handling
- String, array, and pointer sorts
- Search directories and archives

Create dazzling screens in text, CGA, EGA, VGA, and Hercules graphics modes with windows that can overlay one another and be moved and resized on the fly. Store megabytes of string, data, or screen snapshots in extended or EMS memory. Draw complex text or graphic screens to memory and snap them on in an eyeblink. The **ProBas** file I/O routines allow you to read or write huge chunks of data at a clip, far faster than with BASIC.

ProBas also has over 300 other essential services, including handy string, date, time, directory, and array manipulation routines; string, screen, and data compression routines; valuable equipment and input routines; and faster replacements for many BASIC commands.

Whether you are a professional or a novice, **ProBas** will boost your BASIC in ways you never thought possible. **ProBas** allows the professional to write faster, tighter code in much less time and allows novices to quickly and easily write professional-quality programs that would be impossible with BASIC alone. The bottom line is **ProBas** adds power and saves time. After all, how much is a few hundred hours of your time really worth?

For all DOS versions of QuickBASIC and BASCOM. ***Just $145.00!***

ProRef™ On-Line Help For ProBas

ProRef provides on-line help for the routines in the **ProBas** library. This hypertext manual links directly to the QB Advisor in QuickBASIC 4.5 so that the **ProBas** reference becomes an integral part of your QuickBASIC on-line manual. Includes information and examples on **ProBas** routines and helpful hints on programming in BASIC. ***Just $50.00!***

ProScreen™ Screen Management

ProScreen is a full-featured screen generator/editor that will save you more design and coding time than you ever thought possible. **ProScreen** treats screens like a word processor treats text to provide complete control over characters, colors, and placement. Design input screens with up to 130 fields, 19 predefined and 2 user-defined masks. Save screens to screen files or .OBJ files and use the tight BASIC/Assembly code that comes with **ProScreen** to access the screens. There's no kludgy code generator here! Access hundreds of input screens with less than 25k of total code. ***Just $99.00!***

ProMath™ Mathematics Library

ProMath is a collection of over 150 high-level routines that provide mathematical functions and operations for programmers who often work in mathematics, science, or engineering. Complex variables, real and complex matrices, real and complex trigonometric and hyperbolic functions and their inverses, solution of linear equations, integration, differential equations, Fast Fourier transforms, graphics support, and many other useful routines are provided.

For years Fortran has been the language of choice for scientific and engineering applications, but it lacks many of the useful features of QuickBASIC. **ProMath** contains most of the Fortran mathematical and numeric functions and allows you to easily translate Fortran code to BASIC or write new programs in BASIC, while retaining Fortran's numerical prowess.

The **ProMath** manual is over 200 pages and provides a complete description of each routine, including any algorithm and the mathematical formula the routine uses, shown in standard notation. For QuickBASIC 4 and BASCOM 6 only. ***Just $99.00!***

ProBas™ ToolKit

The **ToolKit** is a collection of high-level BASIC and assembly modules that use the routines in the **ProBas** library to save you even more hours of grunt work. Why spend hundreds of hours re-inventing the wheel when you can just plug in **ToolKit** modules like:

- Super-fast B-Tree indexing
- Ring, Bar, Pop-Up, Pull-Down menus
- Scroll-bar tag windows
- Dialog boxes with radio buttons
- Two mini-editors with word wrap
- BCD math routines
- Julian date & calendar routines
- Patch .EXE files
- Protected memory storage area

The **ToolKit** now supports EGA and VGA graphics modes for menus, windows, editors, calendars, and more. Complete with BASIC source code and an all-new comprehensive manual. The **ToolKit** requires the **ProBas** library and helps conserve your greatest asset of all—time! ***Just $99.00!***

ProBas™ TeleComm ToolKit

The **ProBas TeleComm ToolKit** is a collection of high-level communications modules that you plug into your code to provide popular file transfer protocols, terminal emulations, login scripts and baud rates up to 115,200 baud. You get:

- Xmodem/Modem7/Xmodem-1k
- Ymodem (single and batch)
- CRC-16 and Checksum
- VT52, VT100, ANSI, BBS etc.
- Auto Dialer & data base
- Documented BASIC source

Why use clumsy SHELLs to complex terminal programs when you can plug just the communications routines you need into your code? Implement just the features and commands you want. Requires **ProBas**. ***Just $75.00!***

Our thirty-day, money-back guarantee assures you the highest quality and our technical support staff is always ready to help.

HAMMERLY
COMPUTER SERVICES, INC.
9309 JASMINE COURT • LAUREL, MD 20707
(800) 343-7484
INT'L ORDERS: (301) 953-2191 FAX: (301) 725-8147
BBS: (301) 953-7730

Add $5.00 per item ($8.00 Canada) for shipping per order. Europe: $39.00 for 1st item plus $5.00 for each additional item. Visa, M/C, C.O.D. (US Only) checks and approved POs accepted. Trademarks **ProBas**, **ProRef**, **ProScreen**, **ProMath**: Hammerly Computer Services, Inc. QuickBASIC, BASCOM: Microsoft Corp.

PROGRAMMER'S PARADISE, INC.
ELECTRONIC REPRODUCTION PROHIBITED

modified to accept additional parameters for the string lengths when called from C to save that added overhead. Indeed, BASIC's use of string descriptors is one of the primary reasons it is so fast at string operations.

I believe it is important to point out how fast and well designed BASIC really is. Many programmers unfairly criticize BASIC as a language solely because of the interpreter that comes with DOS. Besides being extremely easy to use, current versions of QuickBASIC offer a wealth of structured programming features while offering more commands and features than any other high-level language. And, as we have seen, QuickBASIC is also aptly named. PJ

Ethan Winer is president of Crescent Software and the author of QuickPak Professional and P.D.Q. He may be reached at 11 Grandview Avenue, Stamford, CT 06905; by phone at 203-846-2500; on CompuServe as 72657,3070; and on MCI mail as EWINER.

Code follows.

Code from "Fast String-Searching . . . ," by Ethan Winer

```
LISTING 1
;------ SearchString - reveals Microsoft QuickBASIC's INSTR algorithm

;syntax:  Found = SearchString%(BYVAL StartChar%, Source$, Search$)
; where:
;   StartChar% specifies where in the source string searching is to begin,
;   Source$ is the string being searched, and Search$ is the string to find.
;   Found then receives the position in Source$ where Search$ is found
;   (1-based), or zero if there was no match.

SearchString Proc Uses SI DI, Start:Word, Source:Word, Search:Word

    Cld                     ;insure that string instructions are forward
    Push DS                 ;assign ES=DS
    Pop  ES
    Mov  AX,Start           ;put Start% into AX

    Mov  SI,Search          ;get the address for the Search$ descriptor
    Mov  DX,[SI]            ;put its length into DX
    Mov  SI,[SI+02]         ;and its address into SI
    Dec  DX                 ;we'll handle the first character using Scasb
    Js   Exit               ;they slipped us a null string, so return Start%

    Mov  DI,Source          ;get the address for the Source$ descriptor
    Mov  CX,[DI]            ;put its length into CX
    Mov  DI,[DI+02]         ;and its address into DI
    Mov  BX,DI              ;save that in BX to see how far we searched later

    Dec  AX                 ;adjust Start% so 1st character = 0 offset
    Js   NotFound           ;Start% was zero or negative, get out and return 0
    Add  DI,AX              ;advance Start% bytes into the string
    Sub  CX,AX              ;and consider that many fewer characters to search
    Jbe  NotFound           ;they tried to start past the end

    Lodsb                   ;get and skip over the first character in Search$

Scan:
    Repne Scasb             ;find the first character in Source$ that matches
    Jne  NotFound           ;if it's not there, the complete string isn't either
    Cmp  CX,DX              ;are we less than LEN(Search$) bytes from the end?
    Jb   NotFound           ;yes, so there's no point in looking further
    Or   DX,DX              ;was the string only one character long?
    Jz   Found              ;yes, so this must be it

    Push SI                 ;save the current scanning context
    Push DI
    Push CX
    Mov  CX,DX              ;search LEN(Search$) characters
    Repe Cmpsb              ;compare the two strings
    Pop  CX                 ;restore the context in case we have to scan again
    Pop  DI
    Pop  SI
    Jne  Scan               ;we didn't find it, keep trying

Found:                      ;we found a match
    Sub  DI,BX              ;calculate how far into the string we found a match
    Mov  AX,DI              ;leave the result in AX for the function output

Exit:
    Ret                     ;return to caller

NotFound:
    Xor  AX,AX              ;return zero to show we didn't find it
    Jmp  Short Exit         ;and exit

SearchString Endp
End
```

Unleash the BASIC Power of Hypertext!

Our latest BASIC-booster includes the first hypertext engine designed to be called from BASIC. With the **ProBas HyperHelp ToolKit** you can use the hypertext engine to put one or more manuals on-line with full hypertext search facilities. Imagine having one or more manuals on-line with all of the fantastic hypertext abilities in the QuickBASIC 4.5 compiler and then some:

- Choose single- or multi-window display
- Specify window colors and placement
- Move and resize windows with mouse or keys
- Sophisticated mouse and keyboard interface
- Up to 40 bookmarks to move between hyperlinks

In less than a dozen lines of code you can pop-up context-sensitive help at any time. Your users can then jump to related help, examples, or just browse the manual(s) in one or more windows that they can move or resize at will. The text will automatically wrap within the window to stay fully visible!

Create HyperCard Applications

Use the **HyperHelp** engine to create full-blown hypercard applications. Create sophisticated multi-window, multi-thread hypercard stacks. Mix cards and manuals for total data integration. Moving from link-to-link or card-to-card is instantaneous and the speed will amaze you.

Adding hypertext to your applications takes less than about a dozen lines of BASIC code. Converting ASCII text to hypertext is just as easy—just put delimiters around keywords, hyperlinks, and items you want to display in boldface.

Multiple Help Subsystems

The hypertext engine is just a part of the **HyperHelp ToolKit**. There is a wide selection of help subsystems with various displays, user interfaces, and memory requirements to suit almost any need. Want a small, window-oriented, keyword help system? It's in there! Need a Terminate & Stay Resident help system? It's in there! How about a lightbar indexing system that then pops-up the selected text? It's in there!

The **HyperHelp ToolKit** gets it's blinding speed by using the low-level routine in our **ProBas** Professional Basic Programming Library. **HyperHelp** requires the **ProBas** Library. See our ad on the opposite page for information on **ProBas**, shipping rates, and our thirty-day money-back guarantee.

Special Introductory Price—For a short time this powerful hypertext engine and collection of help subsystems is available for ***just $99.00!***

HAMMERLY
COMPUTER SERVICES, INC.
9309 JASMINE COURT • LAUREL, MD 20707
(800) 343-7484
INT'L. ORDERS: (301) 953-2191 FAX: (301) 725-8147
BBS: (301) 953-7738

Add $5.00 per item ($8.00 Canada) for shipping per order. Europe: $39.00 for 1st item plus $5.00 for each additional item. Visa, M/C, C.O.D. (US Only) checks and approved POs accepted. Trademarks **ProBas, ProRef, ProScreen ProMath**: Hammerly Computer Services, Inc. QuickBASIC, BASCOM: Microsoft Corp.

Number **293** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Prototyping/User Interface Tools

Proteus
Helios
P.O. Box 22869
Seattle, WA 98122
(800) 634-9986 (206) 324-7208
$199

I have found prototyping tools to be very useful for trying out different ideas for user interface designs. Helios has just released version 4.5 of Proteus, their integrated development system for building prototypes and demos. Proteus is a powerful prototype/demo system designed to be easy to use. Commands use pull-down menus to simplify the learning process, with hot keys also available to provide instant access to most commands. Context-sensitive help (over 150 screen pages) is also available to instantly find the meaning of a command.

Proteus, like most tools in this genre, is designed to easily handle text demos on the PC. However, this latest version adds powerful support for bit-mapped graphics including full-screen images and cutouts for the CGA, EGA, VGA, Super-VGA (800-by-600, 256 color), and Hercules adapters along with mouse events. Proteus can use images in the popular PC Paintbrush PCX or PCC format and provides utilities to snap graphics or text screens from other applications and convert text screens to CGA or EGA PCX files. The integrated development system also allows the user to shell out to his or her favorite paint program to edit these external PCX files. When Proteus shells out, it reduces its memory overhead to less than 3K, leaving plenty of room for PC Paintbrush or your favorite word processor. The system also supports 22 video effects that can be painted at variable speeds for animation effects. In addition to graphics, Proteus has good support for music and custom sounds.

The core of Proteus is a powerful screen painter for composing slides and merging screens captured from other programs. All of the PC extended-character set can be easily entered directly from the keyboard. Lines and boxes are easily drawn with the cursor. Both marked blocks and regions are supported. Block commands include Copy, Move, Fill, Edit, Border, Size, Rotate, Grid and Center text. The screen painter also supports Fill, Edit, and Border manipulations of irregular areas (regions). A convenient Undo feature allows going back one command. Fields for data entry can be defined of type Character, Date, Time, Logical, Integer, Real, and Currency. In fact, Proteus has developed a Form Definition Protocol (FDP) that is becoming supported by a number of DOS screen library vendors as a standard import/export format for on-screen forms.

Proteus supports extensive control logic for running a prototype or demo. Actions can be triggered by keystrokes and key recognition, data-field matches, mouse clicks, time delays, or event combinations. Control actions support subroutines, EXECing to other programs and DOS, sound, and chaining demos as well as moving to another slide. To test your demo, a debug mode is provided. A Proteus run-time version is provided for free distribution of your demos or prototypes. Several PCX and screen-grab utilities are provided.

Proteus comes with an excellent 250-plus-page, indexed manual. The manual has a well-written tutorial section that walks the user through the process of creating a demo. The product comes with a 30-day return guarantee. I found Proteus to provide a thoughtful environment for designing dynamite demos and prototypes for applications.

Mini-Case Tools

Logic Gem Version 1.5
Sterling Castle
702 Washington St., Suite 174
Marina del Rey, CA 90292
(800) 722-7853, (800) 323-6406 in Cal.
$99

Sterling Castle has released version 1.5 of Logic Gem, a DOS Logic Processor and Code Generator. Logic Gem represents one of the new breed of DOS case tools. These smaller case tools allow DOS programmers to assemble from various suppliers a powerful development and debugging environment tailored to their specific needs and tastes.

I have admired this unique software tool since I first tried it last fall at COMDEX. But I was dissatisfied with several details of the program's user interface and configuration options that I thought should be cleaned up. With the release of Version 1.5, a number of my user interface concerns have been addressed, and so I can provide more wholehearted support to this one-of-a-kind and powerful software case tool.

Logic Gem is a logic processor/compiler that allows testing those complex predicates that programmers are so likely to use. The integrated software tool provides a powerful logic editor, interpreter, and source code generator used to construct and test decision-logic tables. Logic tables are created similarly to entering data into a spreadsheet. The strength of Logic Gem is the ability to fully test and debug these logic tables quickly and automatically and then generate source code from the logic tables in your favorite language.

Logic tables may be made up of binary (yes/no) logic conditions or mixed entry (multiple choice—if 'X' is greater than, less than, or equal to) logic. The core of this software tool is the logic editor that allows for easy creation and testing of logic tables. Using these tables, Logic Gem automatically determines contradictory or redundant rules. These logic tables can also be used for documenting software products, both under development and completed.

After completing the logic entry and automated testing of a decision table, the user may choose to generate source code in a variety of languages. The Logic Gem code generator supports C, interpreted and the newer structured BASICs, FORTRAN, Pascal, dBASE, and an English natural-language output that may be used for documentation. Even for languages currently not supported (e.g., ADA), this natural-language output can be used as a model to generate source code quickly.

Using a Lotus-style menu interface, the user can maneuver around the Logic Gem environment for editing, interpreting, source code generation, file-handling options, and other built-in features. This new version now supports a number of hot keys and better cursor movement and editing when constructing decision tables. The menu interface and various configuration-file options provide the user with control over the code generator, including indentation and tab spacing, begin and end blocks, code structure, screen color, and error beeping. The likelihood of various logic conditions can also be used to generate source code for the most common events at the highest point in the source code tree hierarchy. A logic interpreter is also integrated to aid as a debugging tool for walking through program logic.

Logic Gem comes with a competent 200-plus-page manual with a decent index. The manual includes a tutorial on logic tables, the Logic Gem reference manual, a glossary, and various appendices. The Logic Gem internal limits (such as 1024 maximum rules) are fully documented. Logic table files are stored as ASCII text, and the details of the file format are available for use with other programs. Logic Gem excels at rationalizing and debugging complex and nested if/then/else predicates so common with today's structured programming. A few items are still on my wish list, but I can heartily recommend this excellent tool for your current and next programming project. The new full version is very reasonably priced at $99, with a special single-language version (English is always included) available for $49. The software comes with a money-back guarantee, which you're unlikely to use.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Books

***System BIOS for IBM PC/XT/AT Computers and Compatibles, CBIOS for IBM PS/2 Computers and Compatibles* and *ABIOS for IBM PS/2 Computers and Compatibles*, by Phoenix Technologies Ltd.**
Addison-Wesley Publishing Company
1 Jacob Way
Reading, MA 01867
800-447-2226
$26.95 each

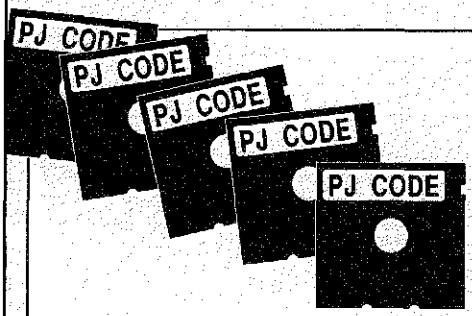
Addison Wesley has just published the first three books in a new series authored by Phoenix, the company that developed the art of "clean room" cloning the IBM PC ROM BIOS and other software. The Phoenix ROM BIOS is currently used in over 14 million PCs worldwide.

Likely to become the most popular book, the IBM PC/XT/AT book (556 pages) focuses on the Phoenix ROM BIOS used in the IBM PC and AT. The other two books provide detailed information on the Phoenix ROM BIOS developed for the IBM PS/2. The CBIOS title (544 pages) covers the part of the PS/2 ROM BIOS designated as the Compatibility BIOS—compatible with the older PC and AT single-tasking operating systems such as DOS. The ABIOS title (688 pages) covers the PS/2 BIOS that supports multitasking operating systems such as OS/2.

The three volumes are similar in style and organization. The first section of each book provides a brief general introduction to BIOS concepts and services, followed by a section on the hardware environment that provides some very useful tables on CPU ports, memory map, and interrupt-table usage. Chapters follow on RAM data usage by the BIOS, system, and CMOS RAM (AT and PS/2).

The remainder of each volume consists of chapters providing detailed information on BIOS services and functions. For the single-tasking ROM BIOS, this consists of chapters covering all of the interrupt vectors used by the ROM BIOS. For the PS/2, the ABIOS data structures, initialization, calling conventions, and extensions are well described. The POST (Power-On Self-Test) sequence is detailed. Appendices cover BIOS error return and beep codes, while the single-tasking books also cover other software details of the hardware environment, keyboard/mouse controller details, and so on.

Unlike their *IBM Technical Reference* counterparts, these books provide details on *undocumented* as well as documented BIOS features. They are not a replacement for a good hardware reference manual for your machine of choice, but at $26.95 for each book (which should be available from your local bookseller), these are quite a bit cheaper and more convenient than the official *IBM BIOS Technical Reference*. PJ



PJ Code Is Available!

On disk, or via BBS.

On disk from:

Programmer's Journal
P. O. Box 30160
Eugene, OR 97403
800-234-0386
503-747-0800 in Oregon
All Code Disks $10.00 each.
To order, see ad on page 96.
PJ Code DiskScriptions are available for only $39.95.
To order, see ad on page 8.

On BBS from:

EDI
1332 Hermosa Ave. #7
Hermosa Beach, CA 90254
213-374-3478
Call for your local access number.

Generation Five BBS
(Voice) 301-495-6690
(Data) 301-495-2932
Accessible via PC Pursuit, Washington D.C. From Files submenu, select "P" for Personal Downloads.

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8868

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
Mention *PJ* when you call.

Logical Alternatives Incorporated
Calder Square, P.O. Box 10674
State College, PA 16805-0674
(Voice) 814-234-8088
(Data) 814-234-1881

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

Soft Advances BBS
10811 Washington Blvd. #205
Culver City, CA 90232
(Voice) 213-559-7105
(Data) 213-559-1449

Save 30 Hours

... the first 3 months you use **Logic Gem** or send it back for a full refund! Less than 1% of our users have returned it. **The other 99% use it often and love it!**

Jerry Pournelle says, *"It has already saved me several hours, and I haven't had it a week. Highly recommended."* (Chaos Manor, BYTE, March 1989)

Logic Gem is completely different from any other programmer's tool. It's all software, requires no special systems or boards. **Best of all, the single language edition is now only $49.95.** And the multi-language edition is only $99.

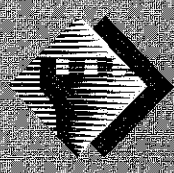
Or try it first for **free**: Down-load a demo of **Logic Gem** from our electronic bulletin board. You'll love it!

Logic Gem, a proven programmer's tool, helps you avoid bugs in three ways:

- **Catches logical errors before you code the program.** Saves hours of debugging time.
- **Automatically generates flawless code** for the "guts" of your program... in C, BASIC, Pascal, FORTRAN, and dBASE.
- **Automatically produces written documentation** of your logic, which insures good communications between designer and coder. (And helps remind you of the logic from one work session to the next.)

Logic Gem works with whatever compiler you are using. The only change with **Logic Gem** you catch and correct the logic bugs before you write the program.

Avoid hours and days of needless debugging time. Call **1-800-722-7853** to put **Logic Gem** to work for you immediately. Order a copy now. Only **$49.95** for a single-language edition or $99 for the multi-language edition. You can return it for any reason within 90 days for a complete refund.



Sterling Castle, Inc.
702 Washington St. #174
Marina del Rey, CA 90292
1-800-722-7853
1-800-323-6406 (in CA)
1-213-306-3020
1-213-821-8122 FAX

See us at Comdex - B1149

Download demo

(213)453-7705 • 3/12/24,8,N,1

Number **268** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVERLOCK

COPY PROTECTION

Designed for clone compatibility & strength

- Thwarts ALL Bit-copy Software
- Protect any COM/EXE w/o Source changes
- Shut down Debug Tracing & Disassemblers
- Supports all Hard & floppy Disk formats
- Restrict or allow Network Installs
- Limit program use by Date or #Executes
- Remotely reset Installs, Date or #Executes
- No damaged media or I/O plugs

For IBM & clones–$195 & up–Free info & Demo

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Number 51 on the Reader Service Card

EVERTRAK

SECURITY

If you HATE Copy Protection, but still want to protect your program from hackers and reverse engineering, **EVERTRAK** allows you to:

- Protect any COM/EXE w/o Source changes
- Shut down DEBUG systems and disassemblers
- Embed/display hacker-proof Customer information
- Restrict use by date-range and/or media-type
- Track unauthorized copies back to their source
- Totally invisible to End Users

EVERTRAK supports IBM & Compatibles using DOS 2.0 or later, sells for $295.00 and has a 30 day money back guarantee. Free info & demo.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Number 310 on the Reader Service Card

You've seen Michael Abrash — "ON GRAPHICS . . ." Now get the book!

POWER GRAPHICS PROGRAMMING

by Michael Abrash

Recently released by ***Programmer's Journal*** in cooperation with **Que Corporation**, this book is sure to become required reading for anyone programming the EGA or VGA.

This is the complete collection of articles from the popular **"On Graphics"** EGA/VGA programming series. This volume contains all of the articles as published in ***PJ***, complete with code listings. And all listings are available on disk, so you can easily see the code running on your own system!

Order your copy of Michael Abrash's *Power Graphics Programming* directly from ***Programmer's Journal.***

Power Graphics Programming *$24.95*
Power Graphics Programming Code Disk . *$14.95*

For fastest service, just use your **Visa/Mastercard.**

Call toll-free today— 800-234-0386

Offer good in U.S only. Call for foreign prices.

PASCAL OBJECT LIBRARIES
For Turbo 5.5 & Quick Pascal 1.0

Andrews Data Systems introduces OOP tools for Pascal. Prototype sophisticated user interfaces quickly and easily. Create complex simulations and real-time systems in a fraction of the time required for procedural languages.

Windows - Form/field support and real-time extensions (lifeline and animated bar graphs). Text and graphic mode support. Includes form builder and attribute select.

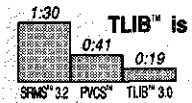
Remote Object Support - distributed processing over serial line or network. Includes serial library.

Concurrent Object Support - multi-task within your application.

$59 each or $89 with source. Serial library alone: $39 or $59 w/source. Windows/Remote/Concurrent: $149 or $199 with source. Add $5 S/H all orders. Check, MO, VISA/MC. No CODs.

ANDREWS DATA SYSTEMS
P.O. BOX 37123
DENVER, CO 80237
1-800-255-2550 ext. 615

Number 318 on the Reader Service Card



Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. SRMS and TLIB 4.10 are later versions.

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Vallino, **PC Tech J**

- **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, Sperry. Integrated Landon Dyer PD **MAKE.**

MS-DOS 2.x, 3.x, 4.x ***Just $99.95*** + $5 s/h Visa/MC
5 station LAN license $299.95 + $5 s/h, call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 (919) 856-0475

Number 288 on the Reader Service Card

C AND RS-232 COMMUNICATIONS

LIBRARY WHICH SUPPORTS USE OF STANDARD ASYNCHRONOUS COMMUNICATIONS ADAPTOR WITH MSC 4.x, 5.x, AND 80x86

Library provides routines which install an interrupt handler that fills wraparound buffer as characters are received. Supports all possible baud rates, both/either COM1 & COM2. Designed to ensure ease of installation and provide highly reliable, low overhead use of asynchronous communications adaptor.

Package contains: routine library, software examples in C and Assembly, short manual describing asynchronous card and provided routines.

Price: $34.00 ppd USA, PA add 6% tax. Check or Money order please. Satisfaction or full refund (15 days).

Kostraba Engineering Associates
6412 Aylsboro Avenue
Pittsburgh, PA 15217
(412) 421-4409

Number 315 on the Reader Service Card

ADVANCED PROGRAMMER'S BOOKS & TOOLS FOR MS DOS & OS/2

Please call or write to Michael J. Young for a FREE BROCHURE

YOUNG SOFTWARE ENGINEERING
20 Sunnyside Ave.
Suite A
Mill Valley, Ca 94941
415-383-5354

Number 243 on the Reader Service Card

Cut time wasted in lines

. . . with a new simulation tool that allows you to design queuing systems easily. Now you can discover the hidden dynamics of waiting lines and minimize waiting time subject to policies that you determine.

The NQ ("enqueue") simulation tool can help optimize facilities used for public access, factory production, transportation, and cargo. NQ is especially well suited for design of telephone systems because it has built-in provision for multilevel and shared resource queues.

NQ automatically provides 11 detailed statistics showing the percent of busy time, average queue length, and more for each resource.

NQ includes a comprehensive 200+ page manual, complete source code and several working models. $995.

Needs PC or above with 512K; DOS 2.0+; CGA or above graphics adapter.

DAIR Computer Systems
3440 Kenneth Drive
Palo Alto, CA 94303
Tel: (415) 494-7081
FAX: (415) 494-6831
Munich (089) 780-9668
Tokyo: (03) 223-0941

Number 5 on the Reader Service Card

Help Is Here.

Give your product a uniform high-quality look and feel, with a minimum of effort.

ScreenLib™
Complete C Source Code!
No Royalties!

Simple to Program ★ Windows
Pop-up Menus ★ Help
Lots of low-level functions

Introductory Price: **$69.95**
Plus $5.00 s/h ($10.00 outside US)
CA residents please add sales tax

Business Computer Services
1800 S. Robertson, Ste. 206
Los Angeles, CA 90035
VISA/MC: 213/836-5026
Demo disk available—$10.00

Number 286 on the Reader Service Card

Network API Library

For *C* or *Pascal*, supports both *Microsoft* and *Turbo* compilers, Compatible with *IBM, 3Com, Banyan*, and *Novell* Networks. Includes DOS network functions, **NetBIOS, IPX/SPX, Client/Servers, Remote Procedure Calls,** Remote screens/windows. $99.95

NetWorkStation Utilities

Network Consulting, Distributed Processing, Classroom Screen Broadcasts, Work Groups, Device Sharing. NetBIOS/Netware $99.95

Information Software Associates
Box 2582, Iowa City, IA 52244
Call for more info (319) 354-6804

Number 294 on the Reader Service Card

COLLEGE DEGREES

. . . from fully-accredited Colleges and Universities. We help Computer Professionals avoid years of unnecessary class work. Get the job offers and recognition you deserve. Phone UDA for more information and our free booklet "Career Tactics and Strategies."

University Degree Advisory
☎ 800-765-7272

Number 311 on the Reader Service Card

PRODUCED 2004 BY SANZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Double-decker
STAND
$124.99
Sturdy, tubular steel, enamel finished oak and putty.

Piggybacks two printers. Casters optional- $8.00.
24x22x41¾High

Single printer stands- tubular steel, oak top. * Includes casters. Discounts avail.

132 col. 24x14x26½H	$73.00*
80 col. 16x14x26½H	59.99

CRT workstations/tables avail., call for prices. 214 /**299-6658**

TRI-STAR DISTRIBUTION
3816 Flamingo Ln., Irving, TX 75062

Number **314** on the Reader Service Card

dBASE FILE ACCESS FROM C

Code Base 4 is a library of C routines which gives complete dBASE and Clipper functionality and file compatibility. The routines correspond directly to dBASE commands. Create multi-user applications using pulldown menus, popup windows and data entry.

Code Base 4 supports DOS, SCO Xenix, OS/2 and Microsoft Windows.

$295 with Source! FREE DEMO.

Sequiter Software Inc.
P.O. Box 5659, Station L
Edmonton, Alberta
Canada T6C 4G1
403-439-8171
Fax: 403-433-7460 **RS# 269**

CSOURCE APPLICATION GENERATOR

C code WYSIWYG screen, report, & program generator. Library code may be enhanced for custom generation. Supports multi-user development & DOS 3.1/Novell networks. Produces small sized DATA ENTRY, REPORT, MENU, INDEX REBUILD & FILE RESTRUCTURE programs with multiple screens & files. Very fast development & execution speeds. Requires 256K RAM, DOS 2.1 or later, Faircom Ctree or Novell Btrieve. For Turbo or Microsoft C 5.0. No royalties, ALL generated applications source is supplied.

$395.00 executable, $595.00 full source code. Multi programmer licenses.

MBAC
13320 Castana Ave.
Downey, CA 90242
213-531-7259 **RS# 291**

BTRIEVE ALTERNATIVE

Application programmers: Get a secure, multilanguage, multiuser, full, flexible indexed file manager: AXOS, only $395. "AXOS provides 57 functions, including audit trail and key-only file management functions, and 64 index paths vs. Btrieve's 31 operations and 24 access paths." -*PC TECH JOURNAL* (Oct 1988)

McSoft, Inc.
P.O. Box 12044
Atlanta, GA 30355
404-984-0608 **RS# 230**

WHITNEY EDITOR $39

Small and fast. Uses all available memory. Split screen editing. Configurable keyboard. Regular expression search. One key compile. Features for writing documentation. Condensed/Outline display. Runs on IBM PC's, AT's, and PS/2's.

USA shipping and handling $3; Outside USA $15. CA residents add sales tax.

Whitney Software, Inc.
P.O. Box 4999
Walnut Creek, CA 94596
415-933-9019 **RS# 298**

PJ CODE IS AVAILABLE

ON DISK

$10.00 each, Visa/MC
CALL:
800-234-0386

ON BBS

See ads on pages 54 & 101 for BBS downloads.

DO IT YOURSELF!

XenoCopy-PC
PC-DOS program lets your PC *Read/Write/Format* *over 350 formats* **$79.95**

XenoFont
high quality text screen printouts
$49.95

+$5.00 S/H + Tax if CA.
XenoSoft • (415) 644-9366
2210 Sixth Street,
Berkeley, CA 94710

Number **123** on the Reader Service Card

"QUICK & CLEAN" COBOL

The COBOL Report Activator (COBRA):

- Generates efficient, structured COBOL programs
- Is easy to learn and use
- Accepts your COBOL code and COPY members
- Produces audit trail reports
- Up to 5 output files per program
- Provides report options & site defined defaults
- Is available for Mainframe & PC

COBRA
Systems Software
David Olivadoti, P.O. Box 750817
Petaluma, CA 94975
(707) 763-5123

Number **281** on the Reader Service Card

JAKE™: NATURAL LANGUAGE PROCESSING USING C

Call JAKE Routines To Convert Sentences Into C Data Structures. Easy To Build Application Dictionary.

OUR CUSTOMERS USE JAKE FOR

COMMANDS
Scripts
Process Control Systems
Expert Systems
Command Languages

QUERIES
Hypertext Systems
Help Systems
Ad Hoc Database Queries

JAKE $495. Interactive Demo $10. CALL (408) 438-6922 VISA, MC

EKS English Knowledge Systems, Inc.
5525 Scotts Valley Dr. Suite 22
Scotts Valley, CA 95066

Usable with Microsoft C, Turbo C, OS/2, and Windows. Context-sensitive semantic processing. Easy interface to any application. Mem: 64K.

Number **243** on the Reader Service Card

FREE CATALOG!

LATEST AWARD BIOS
PC/XT, 286, 386

COMPILED BASIC TOOLS
Libraries, Utilities

COMPUTER BOOKS
Graphics, OS/2, Assembly

HARDWARE UPGRADES

(800) 423-3400
KOMPUTERWERK INC.
851 Parkview Blvd.
Pittsburgh, PA 15215

Number **155** on the Reader Service Card

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. Only **$59.95**

ORDER TODAY!

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Number **320** on the Reader Service Card

PCHRT V3.00
HIGH RESOLUTION TIMER TOOLBOX

The definitive answer to execution profiling and embedded timer problems. 30 functions manage 100 software timers with microsecond accuracy. Self calibrating. Extensive and flexible timer reporting. Profiles selected ROM BIOS and MSDOS interrupts.

Supports TP 4 & 5, TC, MSC 5.x and QC 2.0. Libraries, manual, examples, and full source included.

$24.95 ppd USA, elsewhere add $4.00, Mich add 4%. VISA/MC

RYLE DESIGN
P.O. Box 22
Mt. Pleasant, MI 48804
(517) 773-0587

Number **210** on the Reader Service Card

dCURSES

A full implementation of the UNIX Curses Library for MS-DOS.

MSC
Turbo C
Lattice C

Object $69.95
Source $149.95
VISA/Mastercard
Call for FREE Shareware Copy

HIGH ASPECT DEVELOPMENT
9821 South Pulaski
Evergreen Park, IL 60642
(312) 424-5997

Number **313** on the Reader Service Card

MIDNIGHT ENGINEERING™

A new magazine for entrepreneurial hardware and software engineers that will encourage and challenge you to personally develop and market your own products.

- **PRACTICAL ARTICLES**
- **INSIGHTFUL INTERVIEWS**
- **DETAILED PRODUCT REVIEWS**

This is your chance to influence a brand-new magazine. PJ readers are invited to a **FREE** copy of the premiere issue of *Midnight Engineering* by calling or writing us with ideas.

DATArx
111 E. Drake Rd., Suite 7041
Fort Collins, CO 80525
303-491-9092

Number **272** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
305	Acme Software	90
282	Acucobol Inc.	83
322	Allsoft	76
318	Andrews Data Systems	102
30	The Austin Code Works	72
207	Automation Education	76
51	Az-Tech Software, Inc.	102
310	Az-Tech Software #2	102
303	BBI Computer Systems	37
244	Block Island Technologies	76
3	Borland International	C4
288	Burton Systems Software	102
286	Business Computer Services	102
*	CNS, Inc.	29
281	COBRA Systems Software	103
301	Complete Logic Systems	46
277	Covox Inc.	61
83	Crescent Software	11
5	Dair Computer Systems	102
272	DATArx	103
308	Daytron Electronics Inc.	38
229	Disk Software, Inc.	24
316	EDI	C3
241	Emerald City Software	6
243	English Knowledge Systems, Inc.	103
307	FG Soft Company	25
261	Genus Microprogramming	18
*	Gimpel Software	77
323	Glenco Engineering Inc.	42
*	Graphic Software Systems Inc.	20
248	Hammerly Computer Services, Inc.	98
293	Hammerly Computer Services, Inc.	99
42	Harvard Softworks	35
122	Hauppauge Computer Works, Inc.	47
154	HavenTree Software Limited	70
258	Helios Software	19
313	High Aspect Development Corp.	103
294	Information Software Associates	102
317	Island Systems	17
226	Ithaca Street Software, Inc.	21
295	Knowledge Dynamics Corporation	28
155	Komputerwerk Inc.	103
315	Kostraba Engineering Associates	102
63	Laboratory Microsystems Incorporated	37
44	Lahey Computer Systems Inc.	58
246	Logical Alternatives Incorporated	30
271	Lugaru Software Ltd.	65
*	Magma Software Systems	33
223	Marietta Systems Inc.	75
291	MBAC	103
230	McSoft	103
143	Metagraphics Software Corporation	23
325	Metalex Systems, Inc.	97
45	MetaWare Incorporated	93
104	Microway	88
121	MISOSYS, Inc.	36
266	Mix Software	74
94	MMC AD Systems	75
319	Mt. St. Helens Software	7
312	MTD Software Development Co. Ltd.	32
109	Nanosoft	45
278	Netwise	55
302	NTERGAID	19
160	Nu-Mega Technologies	3
16	Opt-Tech Data Processing	31
*	Paradigm Systems	28
208	PC Tech	24
6	The Periscope Company	48
*	***PJ Code DiskScription***	8
279	Pocket Soft, Inc.	84
*	***Programmer's Journal*** Mail Order Store	96
97	Qualstar Corporation	38
252	Quantasm Software	69
147	Quarterdeck Office Systems	C2
20	Rational Systems, Inc.	9
324	Rogue Wave Software Associates	29
210	Ryle Design	103
269	Sequiter Software Inc.	103
165	SLR Systems	67
167	Software Development '90	59
98	The Software Engineering Store	62
198	Software Science, Inc.	81
221	Software Security	5
110	Softway Inc.	49
268	Sterling Castle	101
10	South Mountain Software	39
158	Summit Information Systems, Inc.	53
297	Talking Technology, Inc.	10
289	TechnoJock Software, Inc.	87
166	The Tool Makers	52
314	Tri-Star Distribution	103
276	Tri-Technology Systems Inc.	77
25	TurboPower Software	82
290	Ultra-X	69
309	Underware Inc.	45
311	University Degree Advisory	102
125	V Communications, Inc.	67
287	Vestronix, Inc.	1
304	Visitech Software	25
320	Wenham Software Co.	103
100	Western Wares	75
73	The Whitewater Group	43
298	Whitney Software, Inc.	103
123	XenoSoft	103
168	Young Software Engineering	102

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine. The advertisers with an * prefer to be contacted directly. ***PJ***'s Advertiser Index is provided as a free service, and ***Programmer's Journal*** cannot be held responsible for omissions or errors.

COMING ISSUE—

January/February 1990
Volume 8.1

Networking:
NetBIOS Services
by W. David Schwaderer

Sun NFS and Remote Procedure Calls *by Frank Greco*

Using DOS 3.x File Services under Networks *by Steven Baker*

Operating Systems:
Using the DESQview API
by Todd Hoff

ASM:
Dynamic Storage and Fast File I/O
by John Otken

On Graphics:
Fast Circle and Ellipse Drawing
by Michael Abrash

Programmer's Journal
January/February 1990 Vol. 8.1
Advertising Space Reservation Deadline:
November 3, 1989
Materials Due:
November 17, 1989
Please call ***Programmer's Journal*** to reserve your space.
(503) 747-0800

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EARN ROYALTIES NOW

ON THE SHAREWARE YOU'VE DEVELOPED

On October 1st a new on-line service was born. Called EDI, it is an on-line service where shareware enthusiasts go to download shareware. And it is a place where shareware authors, like you, can go to distribute the shareware you've developed.

Unlike other shareware distributors, EDI provides royalties to its authors. Royalties are provided every time a copy of your software is downloaded and every time anyone enters your optional on-line customer support section. Every time.

To attract huge numbers of shareware enthusiasts to EDI (and earn you more royalties), we provide shareware enthusiasts with access to EDI over a growing number of on-line networks.

So, no matter what type of share ware product you offer, we'll help you reach enthusiasts all around the country and all around the world.

As a software developer we invite you to take advantage of our services and use it to distribute your shareware products. There is no fee for submitting a product (other than the network connect-time fee). We will gladly review your product. And if it is accepted, we will make your product available to our callers.

So give us a call. And find out how you can start earning royalties now.

Instructions (MSC Network)

Set your modem to 7-N-1 and dial one of the numbers below. Once connected press [return] [return] to get a # prompt and then press .vtt [return] to enter the MCS network. Follow the registration instructions and once on-line press EDI for instant access.

EDI National Access (MSC Network)

ALABAMA
Anniston (205)820-7609
Birmingham (205)254-3120
Montgomery (205)279-0314
ARIZONA
Phoenix (602)233-9592
ARKANSAS
Little Rock (501)376-1535
CALIFORNIA
Burbank (818)954-9161
Corona (714)736-6820
El Monte (818)442-9291
Garden Grove (714)895-4946
Los Angeles (213)772-6971
Monterey (408)899-7855
Mountain View (415)964-8240
Newport Beach (714)895-4946
No. Hollywood (818)786-4111
Sacramento (916)452-7356
San Bernardino (714)888-7611
San Diego (619)293-3620
San Fransisco (415)391-9080
San Jose (408)554-0344
Van Nuys (818)786-4111
COLORADO
Colorado Springs (719)540-8220
Denver (303)893-3803
CONNECTICUT
Hartford (203)728-6766
Milford (203)728-6766
New Haven (203)877-2751
Stamford (203)327-6745
Wilmington (302)571-1433
D.C.
Washington (301)937-0530
Washington (301)937-0722
FLORIDA
Fort Lauderdale (305)462-3104
Jacksonville (904)398-2074
Miami (305)651-2387
Pensacola (904)438-7203
Tampa (813)626-5235
GEORGIA
Atlanta (404)256-9151
Augusta (404)790-5406
Columbus (404)682-1103
HAWAII
Honolulu (808)524-7950
ILLINOIS
Chicago (312)704-0313
Oak Brook (312)572-8550
Oak Brook (312)572-8566
Rock Island (319)324-0042
Springfield (217)525-6683
INDIANA
Indianapolis (317)634-4301
IOWA
Davenport (319)324-0042
Des Moines (515)224-0011
KENTUCKY
Louisville (502)587-9560
LOUISIANA
New Orleans (504)566-0002
MARYLAND
Baltimore (301)760-1950
MASSACHUSETTS
Boston (617)965-4900
MICHIGAN
Detroit (313)963-9141
Grand Rapids (616)456-6047
MINNESOTA
Minneapolis (612)338-5540
MISSOURI
Kansas City (816)283-3719
St. Louis (314)721-4801
NEBRASKA
Omaha (402)341-2425
NEW JERSEY
Boonton (201)335-2332
Cherryhill (609)235-3220
Hackensack (201)489-9020
Moorestown (609)235-3220
New Brunswick (201)846-9310
Newark (201)564-6300
Parsippany (201)335-2332
Springfield (201)564-6300
Trenton (609)883-5113
Whippany (201)335-2332
NEW YORK
Buffalo (716)896-5410
Long Island (516)481-8028
New York (212)391-1920
New York (212)391-0370
Poughkeepsie (914)454-1808
Purchase (914)694-6842
Syracuse (315)428-1674
NORTH CAROLINA
Charlotte (704)374-0587
Fayetteville (919)436-0145
Raleigh (919)821-7535
NORTH DAKOTA
Fargo (701)237-5282
OHIO
Akron (216)535-6635
Cincinnati (513)241-1446
Cleveland (216)621-3880
Columbus (614)461-8450
OKLAHOMA
Lawton (405)355-0275
Oklahoma City (405)272-9751
OREGON
Portland (503)241-0549
PENNSYLVANIA
Harrisburg (717)236-5241
Philadelphia (215)829-0912
Pittsburgh (412)261-9305
SOUTH CAROLINA
Columbia (803)771-7665
SOUTH DAKOTA
Sioux Falls (605)334-1832
TENNESEE
Memphis (901)521-0529
Nashville (615)292-6801
TEXAS
Austin (512)499-8029
Dallas (214)638-7850
Fort Worth (817)460-3711
Houston (713)981-9797
Kileen (817)532-8980
San Antonio (512)525-1692
UTAH
Salt Lake (801)544-0086
VIRGINIA
Newport News (804)874-5121
Norfolk (804)858-2715
Richmond (804)788-0401
WASHINGTON
Seattle (206)624-3580
Spokane (509)455-9398
WEST VIRGINIA
Charleston (304)345-7505
Parkersburg (304)485-7569
WISCONSIN
Madison (608)255-5857

EDI International Access (MSC Network)

AUSTRALIA
Brisbane 07-252-4132
Brisbane 07-252-4124
Melbourne 03-696-5523
Sydney 02-647-9847
BELGIUM
Brussels 02-648-0710
Brussels 02-647-9847
CANADA
Calgary (403)234-7841
Montreal (514)282-0222
Ottawa (613)563-9085
Toronto (416)438-7570
Vancouver (604)669-4040
DENMARK
Copenhagen 1-822015
FINLAND
Helsinki 92919
FRANCE
Paris 1-43-352327
Paris 1-43-207441
Paris 1-43-350101
GERMANY
Frankfurt 069-6666062
Frankfurt 069-6663062
Frankfurt 069-6666881
Frankfurt 069-6668297
Frankfurt 069-6666886
HONG KONG
Hong Kong 5-202525
Hong Kong 5-202857
ITALY
Milan 2-545-6351
Milan 2-545-8840
Milan 2-545-9145
Milan 2-545-8649
Milan 2-540-0425
Milan 2-545-0620
Milan 2-546-8994
JAPAN
Osaka 6-202-4159
Osaka 6-202-7970
Tokyo 3-234-9251
Tokyo 3-234-8275
MEXICO
Mexico City 905-538-7645
Mexico City 905-538-4252
NETHERLANDS
Amsteiveen 020-417855
Amsteiveen 020-476171
NORWAY
Oslo 02-121496
Oslo 02-121795
SOUTH AFRICA
Durban 031-290277
Johannesburg 011-403-5730
SPAIN
Madrid 34-1-742-8244
Madrid 34-1-742-8210
SWEDEN
Stockholm 8-834095
Stockholm 8-834090
Stockholm 8-764-6595
SWITZERLAND
Bern 031-260931
Bern 031-260787
Bern 031-260691
Geneva 22-798-5756
Geneva 22-798-6364
UNITED KINGDOM
London 01-427-4393
London 01-439-4055

ELECTRIC DISTRIBUTION INTERNATIONAL

EDI is a service of ***Electric Distribution International***
1332 Hermosa Avenue, Hermosa Beach, CA 90254.
For customer support call (800) 776-6076
Outside the continental United States call (213) 374-3478

Number **316** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BORLAND'S TURBO C PROFESSIONAL 2.0

CORPORATE TOP RATED PC WEEK 1989 SATISFACTION POLL

PC WEEK POLL: C COMPILERS

	Overall Weighted Score	Overall Reliability	Complete of Command Descript.	Overall Perform.	Complete & Organiz. Document.	Document Clarity	Compiling Process Efficiency	Product Support Quality	Value Relative To Cost	Product Support Access.
Turbo C 2.0 (Borland International)	81	87	79	84	77	78	86	72	70	93
C Optimizing Compiler 5.1 (Microsoft Corp.)	76	83	80	81	78	74	76	68	67	70
C++ 1.07 (Zortech Inc.)	66	68	64	71	63	63	69	60	58	76

"Microsoft was No. 1, but they have been unseated by Borland." PC Week, *May 8, 1989*

PC WEEK POLL: SOFTWARE DEBUGGERS

	Overall Weighted Score	Overall Reliability	Effective. Programmer Interface	Document. Clarity	Complete. Command Descript.	Complete. & Organize. Document	Overall Perform.	Integration Within Programming Environment	C Compiler Compatibility	Product Support Quality	Product Support Access	Value Relative To Cost
Turbo Debugger 1.0 (Borland International)	84	89	90	81	81	81	89	88	81	73	72	93
Codeview 2.2 (Microsoft Corp.)	73	80	71	72	74	74	74	74	78	67	64	72

"Borland's Debugger outshines Microsoft's Codeview." PC Week, *May 15, 1989*

It's two winners in one.

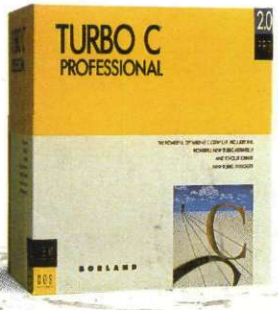
Turbo C,® the core of Turbo C Professional, was the outright winner in *PC Week*'s Poll of Corporate Satisfaction on C compilers. Overall, Borland won with 81. Microsoft® placed second.

Turbo Debugger,® also included in Turbo C Professional, was the outright winner in EVERY category in *PC Week*'s Poll Of Corporate Satisfaction on Debuggers. And, once again, we topped the score with 84, overall. Microsoft came in second-best, 11 points behind.

Get Borland's Turbo C Professional and get the best of both worlds: our top-rated C compiler and our top-rated Debugger.

Call **(800) 345-2888*** and we'll send you both *PC Week* polls and technical specifications on Turbo C and Turbo Debugger.

Turbo C Professional includes both Turbo C 2.0 and Turbo Assembler® & Debugger.



BORLAND

Code: MC41

From Canada, Call (408) 848-4391. Reprinted from PC Week, May 8, 1989; PC Week, May 15, 1989. Copyright ©1989 Ziff Communications Company. Turbo C, Turbo Debugger, and Turbo Assembler are registered trademarks of Borland International. Copyright ©1989 Borland International, Inc. All rights reserved. BI 1328

Number 3 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED